НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 "Системний аналіз"
на тему: «Методи машинного навчання в задачах розпізнавання і класифікацій особливостей документів**»**

Виконав :
Студент IV курсу, групи КА-61
Мрозек Євген Романович _____

Керівник:
Д.ф.м.н., доц. Ігнатенко О. П. _____

Консультант з економічного розділу:
доцент, к.е.н. Шевчук О.А. _____

Консультант з нормоконтролю:
доцент, к.т.н., доцент
Коваленко Анатолій Єпіфанович _____

Рецензент: _____
Д.ф.м.н., академік НАН України,
директор Інституту програмних систем НАНУ
Андон Пилип Іларіонович

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу2

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – **124 "Системний аналіз"**

Освітньо-професійна програма **«Системний аналіз і управління»**

В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Мрозек Євгену Романовичу

1. Тема роботи «Методи машинного навчання в задачах розпізнавання і класифікацій особливостей документів», керівник роботи Д.ф.м.н., доц. Ігнатенко О. П. затверджені наказом по університету від

«25» травня 2020 р. №1143-с

2. Термін подання студентом роботи

3. Вихідні дані до роботи

4. Зміст роботи

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., доцент	20.04.20	04.05.20

7. Дата видачі завдання .

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка

Студент Євген Романович

Керівник Ігнатенко О. П

РЕФЕРАТ

Дипломна робота: 65 с., 37 рис., 9 табл., 2 дод., 13 джерел.

КЛАСИФІКАЦІЯ ДОКУМЕНТІВ, КОМП'ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ПЕРЕДАВАЛЬНЕ НАВЧАННЯ

Мета роботи: навчити нейронну мережу для класифікації з прийнятною точністю при мінімальних об'ємах даних.

Актуальність теми пов'язана з тим, що кількість і типів створених документів росте з кожним днем. Тому необхідні моделі які будуть автоматично перевіряти їх правильність.

Об'єкт дослідження: скан-копії документів при укладенні договорів

Предмет дослідження: нейронна мережа для розпізнавання і класифікацій особливостей документів. В якості підходу навчання нейронної мережі використовується передавальне навчання(transfer learning).

Проведене порівняння звичайного навчання нейронної мережі та навчання з використання підходом передавального навчання в області комп'ютерного зору.

Отримані результати: нейронна мережа, що класифікує документи за кількістю на них печаток.

Abstract

Thesis: 65 pp., 37 Fig., 9 Table., 2Appendix, 13 sources.

CLASSIFICATION OF DOCUMENTS, COMPUTER VISION, MACHINE LEARNING, NEURAL NETWORKS, TRANSFER LEARNING

Purpose: to teach the neural network to classify with acceptable accuracy with minimal dataset.

The relevance of the topic is due to the fact that the number and types of created documents is growing every day. Therefore, models are needed that will automatically check their correctness.

Object of research: scanned copies of documents of contracts

Subject of research: neural network for recognition and classification of features of documents. Transfer learning is used as a neural network learning approach.

A comparison of conventional neural network training and training using the approach of transfer learning in the field of computer vision.

The results obtained: a neural network that classifies documents by the number of seals on them.

Зміст:

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ :

CV - Computer Vision

NN - Neural Network

TL - Transfer Learning

Train - дані предназначені для навчання нейронної мережі

Valid - дані предназначені для перевірки нейронної мережі

CPU - Central processing unit

GPU - Graphics Processing Unit

LR - Learning Rate

CNN - Convolutional Neural Network

ВСТУП

Для того щоб дитині розказати хто така корова необхідно кілька фотографій, щоб вона навчилась їх вирізняти серед інших тварин. Дорослій людині достатньо прочитати про корову для розпізнавання. Сучасні алгоритми машинного навчання дуже жадні до даних. Наприклад датасет MNIST містить в собі 60 тис зображень для 10 класів дуже маленької роздільної здатності 28x28 пікселі. Також до проблем додається необхідність розмічених даних для навчанням з вчителем. Humansintheloop пропонує нам від 1 до 5 центів за кожне зображення. Тобто зібрати дані на 100 тис може коштувати вам 2 тис $. Це добре якщо у вас є не розмічені дані - для деяких областей отримання фотографій може бути дорожче ніж їх розмітка або неможливо їх отримати. Наприклад Західний чорний носоріг вимер у 2011 році і нові справжні фотографії неможливо отримати.

Один з шляхів вирішення цієї проблеми Передавальне навчання (Transfer Learning). TL - це методологія збереження знань під час вирішення однієї проблеми та застосування її до нової, але пов язаної проблеми. TL послаблює гіпотезу, що дані мають бути незалежні та однаково розподілені.

Тому в роботі поставлено для виконання наступну задачу:

1. огляд найсучасніших архітектур NN для задачі класифікації;
2. огляд технології передавального навчання;
3. підготовка маленького набору даних;
4. порівняння ефективності звичайного підходу CV та передавального навчання для класифікації документів;

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ

1.1 Вступ

В данному розділі буде розглянуті основні поняття нейронної мережі для задач комп'ютерного зору. Функції втрат для задач класифікації. Проводиться аналіз шарів нейронної мережі, сучасні мікроархітектура та чому вони використовуються в найсучасніших NN. Перелік сучасних нейронних мереж буде зосереджений на тих, що показали гарні результати на щорічному змаганні по класифікації ImageNet і архітектуру та коефіцієнти яких ми можемо скачати.

1.2 Комп'ютерний зір

Комп'ютерний зір - це галузь машинного навчання призначена, для того щоб на високому рівні взаємодіяти з зображенням або відео. Головна мета автоматизувати процеси, які може виконувати візуальна система людини. Цей напрямок почав розвиватись з 1970 року, але набирає оборотів в останні 10 років. Це в основному пов'язане з розвитком CPU та особливо GPU, адже для CV необхідні величезні потужності. Але не все так радісно як здається. Отримати власний сервер з кількома відеокартами необхідні великі кошти, особливо після скачка цін на них у зв'язку з бумом Біткоіна. Альтернатива - оренда потужностей у Amazona, Google та Microsoft. Але ціни починають з 1$+/год - і за цю ціну інколи краще орендувати кілька потужних CPU.

Так як камера є майже в кожному сучасному телефоні, то інформації і стимулу для обробки зображень достатньо. Подальший розвиток може бути пов'язаний зі зниженням цін на потужності, оптимізації архітектур а також використання технології передавального навчання в області комп'ютерного зору. В своїй роботі буду розглядати саме на останній пункт - TL.

1.3 Нейронна мережа в CV

В чому основна відмінність NN в CV від звичайних, що кожне зображення це ширина*висота пікселів. А кожний піксель зазвичай кодується 3 кольорами RGB. Уявимо - що розміри наших картинок 2x2 пікселя закодованих RGB - тобто кожен піксель це 256^3 унікальних комбінацій. Картинка 2x2 це 256^3*2*2 = 79228162514264337593543950336 можливих комбінацій. Можемо уявити що коли роздільна здатність збільшується - кількість унікальних комбінацій стрімко прямує до нескінченності. Хоча картинки теоретично можуть бути згенеровані, але ж для навчання NN необхідні розмічені дані. Саме тому виходячи з таких особливостей вхідних даних звичайні повнозвязна багатошарові перцептрони не можуть вчитись і давати гарний результат. Саму тому в основі NN в CV лежать шари з набагато менше кількістю параметрів та їх основна задача у виявленні головних ознак зображення.

1.4 Активаційні функції

Активаційна функція нейрону - залежність вихідного сигналу нейрону від вхідного. Коли передавальна функція нелінійна, то, як доведено, двошарова нейронна мережа є універсальною апроксимацією функцій. Метод зворотного поширення помилки вимагає, щоб функція активації була неперервною, нелінійною, монотонно зростаючою, і диференційованою.

Ми будемо використовувати лише ReLU (1.1) [1]:

$$f(x) = \begin{cases} 0, x \leq 0; \\ x, x > 0. \end{cases} \tag{1.1}$$

Та softmax (1.2) на останньому шарі для знаходження ймовірностей:

$$softmax(y_i) = \frac{exp(y_i)}{\sum_i exp(y_i)}, \tag{1.2}$$

де y_i - і-та компонента вектора.

1.5 Класифікація шарів нейронної мережі

Повноз'єднані шари (fully connected layer) з'єднують кожен нейрон одного шару з кожним нейроном наступного шару формула (1.3). Це, в принципі, є тим же, що й традиційна нейронна мережа багатошарового перцептрону рис (1.1).

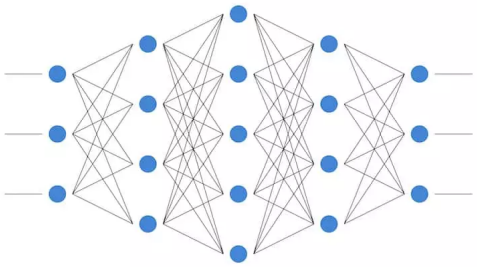


Рисунок 1.1 - багатошаровий персептрон

$$a^{[l+1]} = f(W * a^{[l]} + b), \tag{1.3}$$

де $a^{[l]}$вихід з шару l;

W - ваги;

b - зміщення;

f (x) - активаційна функція.

Dropout - виключення нейрона з ймовірністю p. Це означає що будь який нейрон в зовнішньому або прихованому слої з заданою наперед ймовірністю не буде врахований під час прямого чи оберненого поширення помилки (forward and backward). [2]

На рис (1.2) наведений приклад однакової архітектури NN до і після застосування Dropout

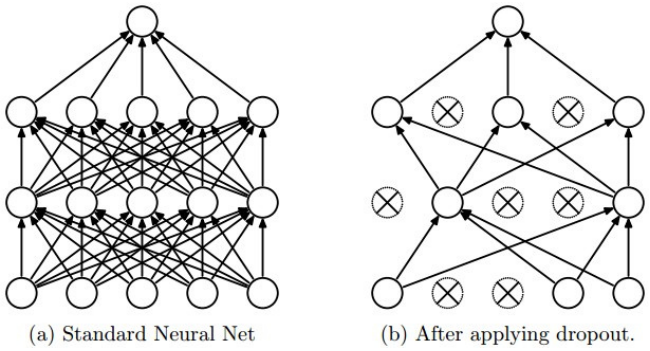


Рисунок 1.2 Ліворуч - нейронна мережа до використання дропауту, праворуч - після

Convolution - це взаємно кореляційна функція. Або як її частіше називають у машинному навчанні - операція згортки. Різниця лише у виконанні властивості асоціативності. Яка не потрібна і лише потребує зайвих обчислень математична формула згортки в дискретному вигляді за формулою (1.4) :

$$(f*g)(t) = \sum_{\tau=-\infty}^{\infty} f(t-\tau)*g(\tau) \qquad (1.4)$$

Рис (1.3) ілюструє особливості згортки при роботі напряму з пікселями зображення.

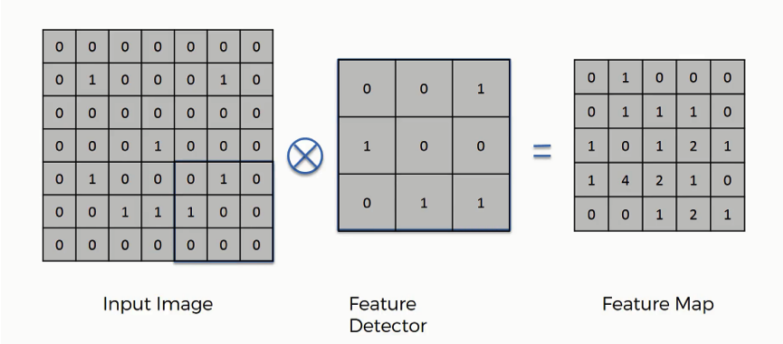


Рисунок 1.3

Так як застосування згортки зменшує розмірність зображення - перед входом в шар добавляють рамку з нулів, позначається pad. Pad = 2 означаєь що з усіх сторін добавили по два шари нулів. Якщо ми використовуємо згортку до

всіх пікселів по черзі, то в цьому випадку stride = 1. Stride - крок з яким ми ходимо по пікселям. Розмір згортки, pad і stride - це все гіперпараметри.

Агрегувальні шари (Pooling) - зменшують розмірність зображення з невеликою втратою інформації. Вони використовують двох видів - Максимальний та середній середній на рис (1.4) .

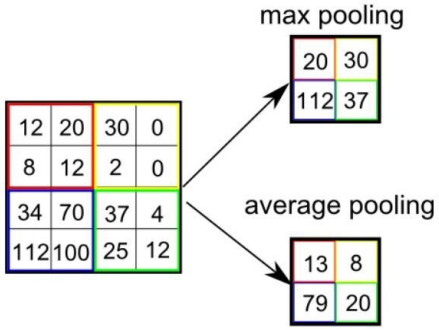


Рисунок 1.4

MaxPolling за формулою (1.5):

$$a^{[l+1]}(i,\ j) = max_{i,j}(\ a^{[l]}(i-n,\ j-n)\ ),\ n\ =\ \underline{-f_c, f_c}\ ,\quad (1.5)$$

де i, j - координати;

f_c - розмір матриці пулінга.

1.6 Функція втрат для CV класифікації

NN навчається за допомогою стохастичного градієнтного спуску і потребують функцію втрат. Наша задача вибрати правильну функцію втрат, щоб зменшення її означало покращення нашої моделі. Максимальна правдоподібність використовується для пошуку найкращих оцінок параметрів моделі для навчальних даних. Модель намагається робити прогнози, які відповідають розподілу даних цільової змінної. І функція втрат в цьому значенні намагається оцінити наскільки схожі розподіли прогнозів та цільової змінної. А також максимальна правдоподібність має властивість консистенності - “більше

даних - кращі оцінки параметрів”. Згідно максимальної правдоподібності похибка між двома розділами - перехресна ентропія.

$$H(p,q) = -\sum_{x \epsilon X} p(x) * log\, q(x), \qquad (1.6)$$

де p,q - ймовірності.

В задачах класифікації хочемо виміряти ймовірності належності до кожного класу. В навчальних данних ми вже знаємо відповідь - це або 0 або 1 (не належить і належить відповідно). Ми намагаємось мінімізувати ентропію між справжніми та спрогнозованими ймовірностями. Останнім шаром моєї нейронної мережі буде персептрон з кількістю нейронів відповідно до кількості унікальних класів та активаційною функцією - softmax. В моєму коді буде sparse_categorical_crossentropy - це для оцінки помилки між [0.1, 0.2, 0.3, 0,2, 0.2] та лейблом 2 (а не [0, 0, 1, 0, 0]).

1.7 Існуючі архітектури

1.7.1 CNN

CNN - це одна з архітектур нейронних мереж, які найчастіше використовують для задачі CV. Особливість цієї нейронної мережі є використання Convolution layer, від чого і пішла назва. CNN являє собою повнозв’язну NN в якій всі нейрони попереднього шару пов'язані з наступним шаром. Ідея лежить в виявленні головних ознак. Це можуть бути вертикальні і горизонтальні кути, якщо використовувати згортки такого виду рис (1.5). [3]

1	1	1
0	0	0
-1	-1	-1

1	0	-1
1	0	-1
1	0	-1

Рисунок 1.5

Але практика показала, що використання коефіцієнтів введені людиною буде неефективним. Доцільніше під час ініціалізації генерувати випадкові ваги, а далі змінювати їх наприклад за допомогою методу зворотного поширення помилки.

Стандартна архітектура полягає у використанні Conv, за яким слідує Polling, знову Conv, Polling за який іде 2 слої FC шару. Якщо задача класифікації в кінці ставиться - то це ще один FC (кількість нейронів = кількості класів) з активаційною функцією softmax, як на рис (1.6). Зараз в більшості випадків використовується MaxPolling, на відміну від початкової версії з Avg Pooling та виходом з активаційною функцією сігмоїди.

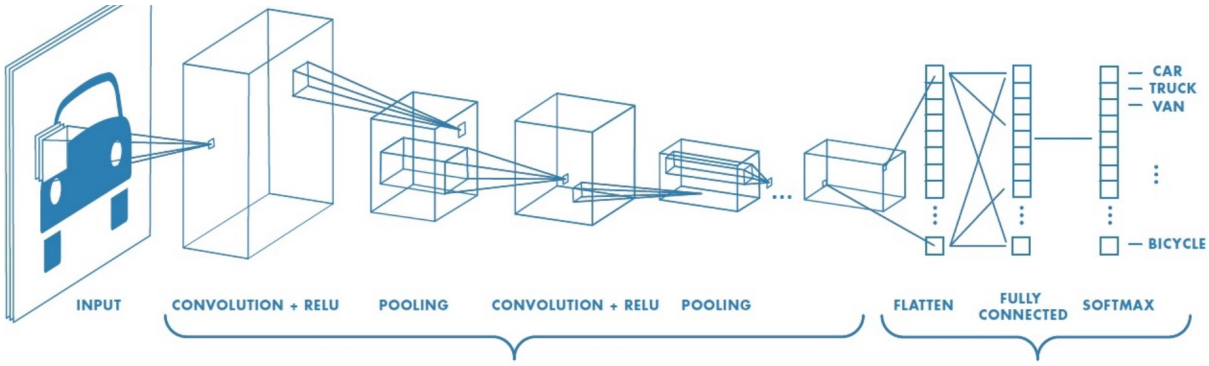


Рисунок 1.6 - стандартна архітектура CNN

1.7.2 VGG

VGG - це нейронна мережа побудована за принципом CNN рис (1.7), але кількість шарів у ній у кілька разів більше. VGG16 і VGG19 це означає що в цих архітектурах 16 і 19 прихованих слоїв відповідно. [4]

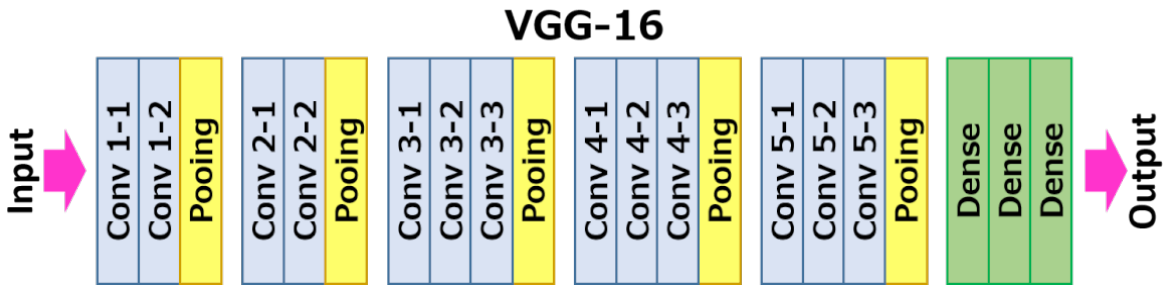


Рисунок 1.7 - архітектура VVG16

Найсучасніші нейронні мережі містять більше 100+ прихованих слоїв, але за кількість параметрів VGG16 і VGG19 навіть сьогодні виглядають гігантами навіть за сьогоднішніми мірками в 110 мільйонів параметрів.

1.7.3 Resnet

Але подальше збільшення глибини в архітектурі не буде давати великого збільшення точності, навіть навпаки. Дослідження показують що точність буде погіршуватися, як на рис (1.8). [5]

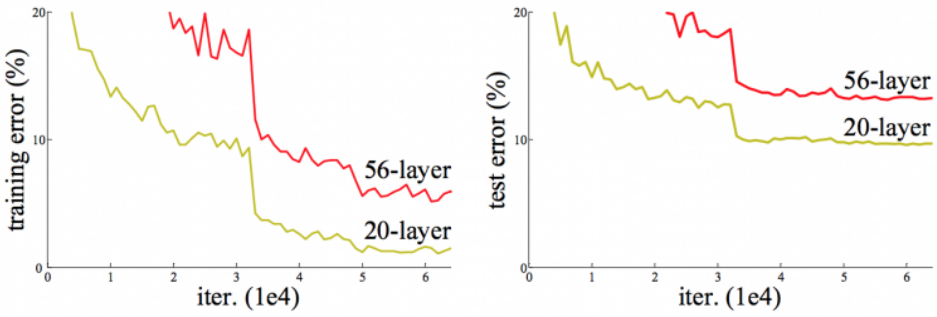


Рисунок 1.8 - точність нейронної мережі на train і valid при різній кількості шарів

Наступним кроком в розвитку CNN стає ResNet. Ідея ResNet полягає в передачі результату одного слою через “Короткий шлях”. Уявимо що у нас нейронна мережа складається з n = 19 шарів, як у VGG19. Ми добавляємо нові слої Conv+Polling і будемо добавляти короткий шлях. Якщо добавлення ще одного шару буде погіршувати точність - ваги цього блоку будуть 0. І ми легко отримаємо функцію ідентичності, як на рис (1.9):

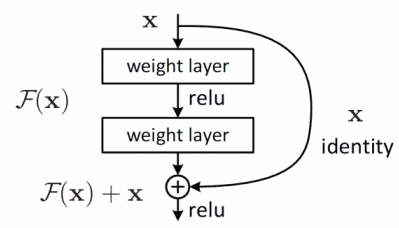


Рисунок 1.9 - Приклад прокидування результату в наступні шари

$$a^{l+1} = g(W*a^l + b + a^l),$$

де a^l - вихід шару l;

W = 0;

b = 0;

g - активаційна функція RELu.

$a^{l+1} = g(0*a^l + 0 + a^l) = g(a^l) = a^l$

Тобто результати однозначно не будуть гірше початкової архітектури на навчальній вибірці. Дослідження показують, що помилка буде зменшуватись,як на рис (1.10).

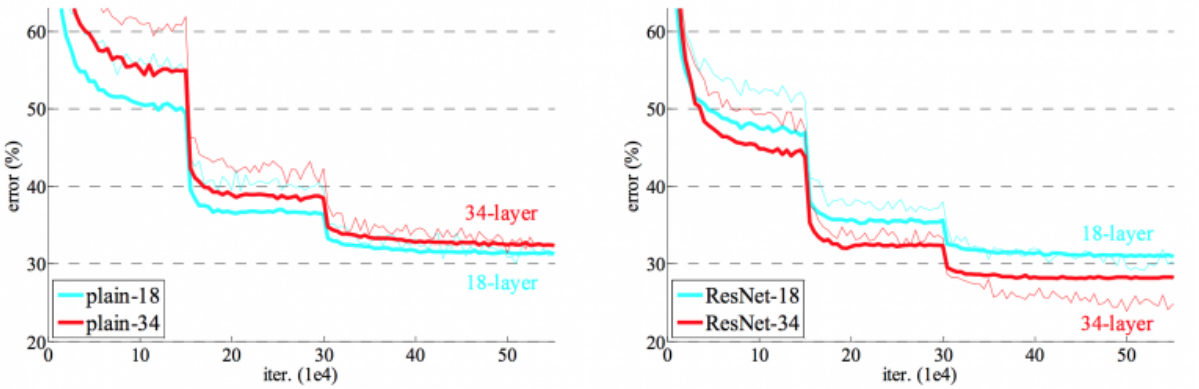


Рисунок 1.10 Точність архітектури ResNet на train і valid при різній кількості шарів

1.7.4 Inception

Основна проблема при побудові власної архітектури - вирішити згортку якого розміру використовувати, 1x1, 3x3 , nxn. Але для кожного окремого завдання використання одного, може бути краще іншого. Що краще два 3x3 чи один 5x5? А чому не використати їх всі одночасно? Це і втілено в архітектурі Inception. На рис (1.11) приведено модуль з якого складається NN. [6]

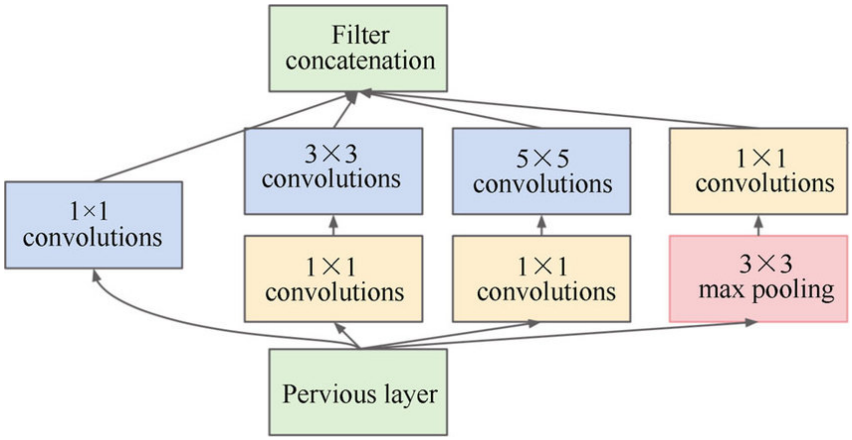


Рисунок 1.11

1x1 Conv - цей шар (Network in Network) дозволяє зменшити кількість операцій для обчислення в 10 разів, порівняно з аналогами без 1x1 Conv. Ось як виглядає фінальна архітектура. [7] Це використання кількох модулів описаного вище в комбінації з Poolling-ами. Як видно на рис (1.12) тут у нас є 3 класифікатора. Це допомагає боротись з перенавчання нейронної мережі.

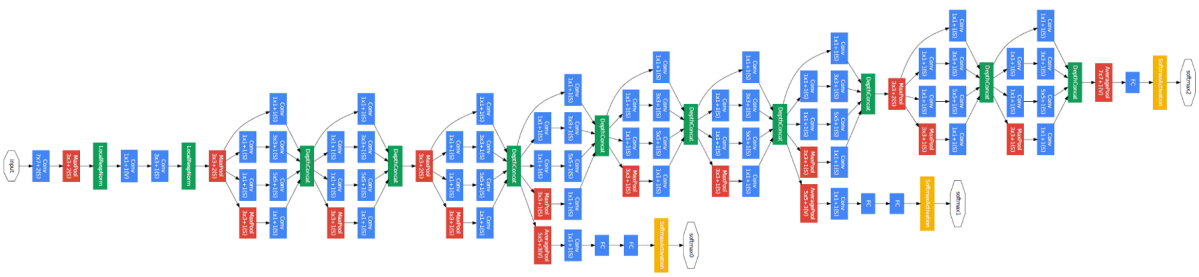


Рисунок 1.12 - архітектура InceptionV3

Обєднуючи ідеї ResNet і Inception - отримуємо Inception ResNet V2 NN глибиною в 572 шари з кількістю параметрів 55M. (Але це вдвічі менше ніж VGG16). [8]

1.7.5 Mobilenet

Mobilenet - так як і в Inception ми багато використовуємо Conv 1x1 рис (1.13), що дозволяє зменшити кількість необхідних обчислень у кілька разів. Також особливість - не використання Polling шарів. Замість цього використовують Conv шари з Stride = 2. Розмірність зображення зменшується зменшується вдвічі, а кількість фільтрів збільшується вдвічі. [9]

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Рисунок 1.13

Є різні варіації MobileNet, але всі вони містять “малу” кількість параметрів. Це зменшує необхідний час, та потужності для навчання та прогнозування. MobileNet V2 може працювати на мобільних телефонах з 5 FPS. MobileNet використовується в YOLO нейронній мережі. [10]

1.8 Висновок до розділу 1

Розглянувши найсучасніші нейронні мережі можемо зробити, що всі вони різні, але мають кілька особливостей. Чим глибше нейронна мережа, тим менше висота і ширина зображення, а число каналів навпаки - збільшується з часом. Банальним збільшенням кількості шарів не можна добитись покращення результату, тому використовують передачу результатів через кілька шарів без обробки.

В рамках передавального навчання ми хочемо використати готову архітектуру разом з їхніми коефіцієнтами. Так у випадку VGG ми можемо взяти всю нейронну мережу, замінивши лише останній шар класифікатора, весь FC або все після будь якого блоку Conv+Polling.

РОЗДІЛ 2 ПЕРЕДАВАЛЬНЕ НАВЧАННЯ

2.1 Вступ:

Більшість навчених NN мають одну спільну особливість: перших кілька шарів зазвичай фільтри Gabora або Blob . Це не пов'язано з завданням і використовуються на великій різноманітній кількості датасетів. Феномен полягає в тому що це не лише при різних датасетах, а навіть при різних задачах: навчання з вчителем, навчання без вчителя. І якщо якась архітектура їх немає - це скоріш за все означає неправильний вибір гіперпараментів. Також ми знаємо, що останні шари унікальні і можуть використовуватись лише для свого конкретного завдання. Наприклад останній шар з активаційною функцією SoftMax. Якби ми більш детально вивчили перехід особливість переходу від загальних фільтрів до індивідуальних - це допомогло б нам зменшити кількість необхідних даних та потужностей для навчання NN.

В цьому розділі ми будемо розглядати кілька питань:

1. перехід відбувається в межах одного шару, чи на протязі кількох?
2. як можна визначити належність шару до загального унікально?
3. наскільки великий вплив має “відстань” між датасетами?
4. краще заморожувати для навчання слої, які ми перейняли з іншого завдання?

2.2 Визначення домену та завдання

Домен будемо позначати як D={**X**, P(X)}, де **X** - простір ознак, а P(X) розподіл ймовірностей, де X = {x_1, …, x_n} ∈ **X**. Завдання може бути представлено як T = {y, f (x)}, де y це цільова ознака, а f(x) цільова функція прогнозування. Також можна розглядати P(y/x) як умовну функцію ймовірності.

Звичайний підхід CV - це коли ми навчаємо нейронну мережу на доменній області D_t і використовуємо для нашого завданняT_t.

TL - коли є завдання T_tно основі домена D_tі ми можемо отримати користь від завдання T_s в доменній області D_s. Завданням TL є покращення результатів прогнозування цільової функції прогнозування f_t(x) для завдання T_tперенесенням знань з D_sта T_s. Тут D_s!= D_tабо T_t!= T_s. До того всього зазвичай доменна область D_sбільша ніж D_t.

Deep Transfer Learning - це випадок, коли наша цільова функція прогнозування не є лінійною функцію (наприклад Нейронна мережа).

Є різні категорії TL в залежності від комбінацій D_s!= D_t; T_t!= T_s; P_s(X) != P_t(X) ; P_s(y/x) !=P_t(y/x); **X**s != **X**t;Y_s != Y_t;

В контексті задачі розпізнавання і класифікації особливостей документів ми зупинимось на варіанті коли:

T_t!= T_s; D_s!= D_t; P_s(X) != P_t(X); P_s(y/x) != P_t(y/x). Немає нейронної мережі навченої на класифікацію особливостей документів, а також немає розмічених даних. **X**s =**X**t (Адже простір пікселів однаковий для двох задач CV);

2.3 Підходи передачі коефіцієнтів

Розглянемо варіанти передачі коефіцієнтів на рис (2.1) та на рис (2.2)

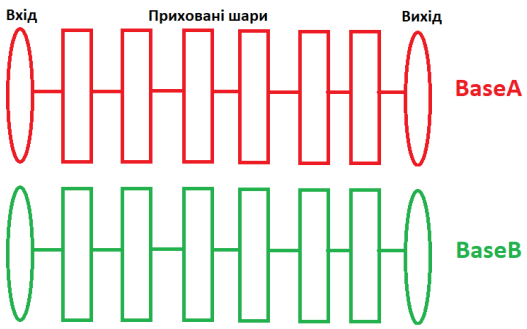


Рисунок 2.1

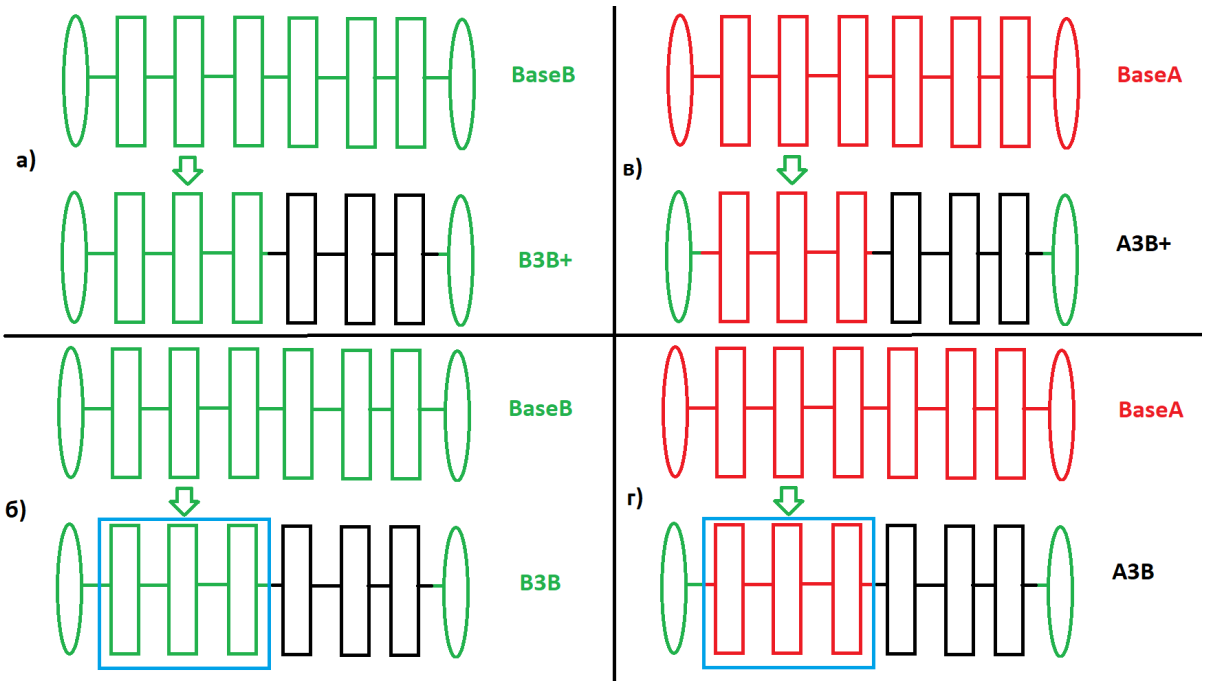


Рисунок 2.2 - підходи передачі коефіцієнтів

У випадку а) Ми навчали мережу на D_b для T_b, далі взяли перших 3 приховних слої і перенесли на нову мережу. Тобто замість випадкової ініціалізації 3 слоїв - ми почала навчання з вже навчених коефіцієнтів.

Випадок б) Ми навчали мережу на D_b для T_b, далі взяли перших 3 приховних слої і перенесли на нову мережу і заморозили для навчання коефіцієнти.

Випадок в) Ми навчали мережу на D_a для T_a, далі взяли перших 3 приховних слої і перенесли на нову мережу. Навчаємо всю нейронну мережу на D_bдля T_b.

Випадок г) Ми навчали мережу на D_a для T_a, далі взяли перших 3 приховних слої і перенесли на нову мережу і заморозили для навчання коефіцієнти . Навчаємо нейронну мережу(без перших 3) на D_bдля T_b.

Умовні позначення цих випадків B3B+, B3B, A3B+, A3B, де 3 - кількість шарів які ми перенесли.

2.4 Перехід загальних фільтрів до унікальних

На рис (2.3) архітектура CNN (AlexNet) яку ми будемо розглядати. Датасет - ImageNet в якому 1000 класів.

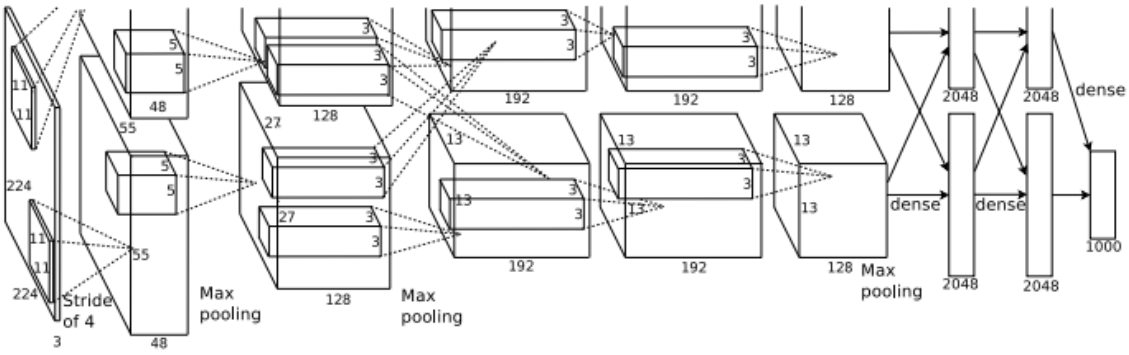


Рисунок 2.3

Ми випадковим чином розбили Весь датасет на 2, в кожному по 500 класів. Навчили NN на A і B, а потім почали використовувати всі техніки описані вище. Розбиття і навчання проходило 10 разів. Точність в експерименті була top1 accuracity, що відповідає звичайній accuracity на рис (2.4). [11]

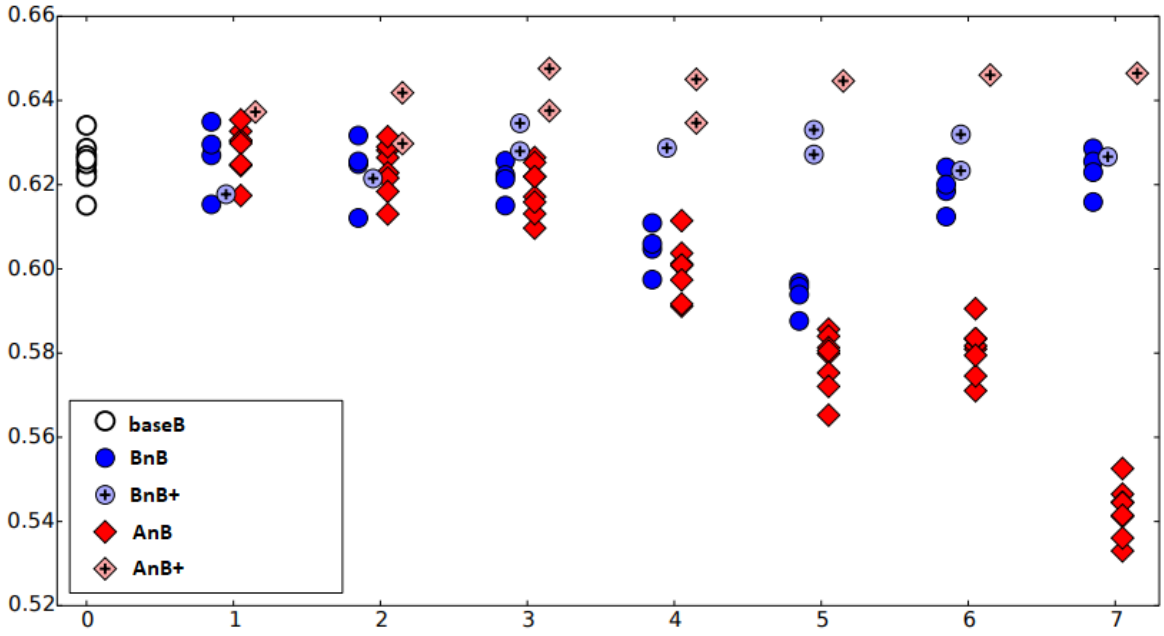


Рисунок 2.4 - точність різних підходів

Тут ми можемо зробити кілька цікавих висновків:

1. BnB - якщо ми візьмемо та перенесемо один шар/два шари - тоність збільшиться/ не зміниться. Якщо n від 3 до 5 включно - зменшиться. Це можна пояснити, що при навчанні градієнтним спуском перші і останні шари взаємодіяли разом. А коли перші 5 шарів заморожені - не може знайти залежності. Але коли $n \geq 6$ уже потрібно набагато менше навчати параметрів і градієнтний спуск легко повертає точність до початкового значення.
2. У випадку BnB+ старт з випадкової ініціалізації. Така точна настройка (fine-tuning) запобігає зниженню точності на всіх шарах.
3. AnB - можемо зробити висновок, що перших 2 шари загальні. 3 ій шар - незначне погіршення. А далі величезне падіння - спостерігаючи точки BnB можемо зробити висновок, що градієнтний спуск не може віднайти залежності, а також шари плавно переходять від загальних до унікальних до кожного завдання. На n= 4,5 домінує ефект перший (Бо різниця між точностямми незначні) Але на n=6,7 домінує другий ефект.
4. AnB+ Спостерігаємо, що точність вище ніж baseB при будь якому n. Раніше вважалось, що цей ефект відбувається якщо доменна область B дуже маленька. Але тут D_b- половина найбільшого розміченого датасету. Також треба сказати, що підвищення точності не відбулось за рахунок збільшення кількості ітерацій навчання порівняно з baseB в два рази. Адже в цьому випадку BnB+ не збільшує свою точність. Підвищення точності можемо пояснити збільшенням узагальненості NN.

В середньому точність AnB+ серед $n = \underline{1,7}$ краще ніж baseB на 1.6%
і на 1.4% краще ніж BnB+.

2.5 Виділення ознак:

В NLP (Natural Language Processing) одна з базових технік це Word2vec. Це спосіб відображення слова в низькорозмірному векторі (embedding). Якщо слова мають схоже значення - їхні вектори будуть схожі. Перетворення слів на embedding часто перший крок в обробці текстів. Така ж логіка і у використання TL. Замість того щоб претренована модель видавала шанси належності до кожного класу - результатом буде перетворення зображення на вектор. А потім уже застосовувати відомі нам техніки машинного навчання. В якось сенсі це схоже на випадок AnB описаний вище.

Simon Kornblith∗ , Jonathon Shlens, and Quoc V. Le у роботі “Do Better ImageNet Models Transfer Better? “ досліджували кореляцію точності моделі на ImageNet та точність передавального навчання цих моделей на 12 найпоширеніших датасетах: object classification (CIFAR-10 , CIFAR-100 , PASCAL VOC 2007 , Caltech-101); fine-grained object classification (Food-101 , Birdsnap , Stanford Cars , FGVC Aircraft , OxfordIIIT Pets); texture classification (DTD); and scene classification (SUN397) на рис (2.5). [12]

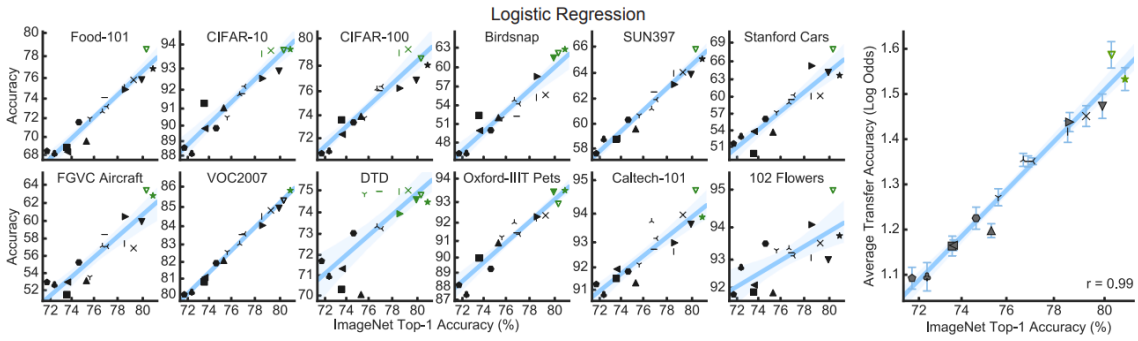


Рисунок 2.5

Якщо на створених embedding запускати Logistic Regression кореляція між точністю на ImageNet та цільових датасетах буде 99%.

2.6 Аугментація даних (data augmentation)

Одним з способів збільшення датасету - аугментація даних. Це методика створення штучних даних за допомогою перетворень зображень навчального датасету або комбінації кілької перетворень: зсув, горизонтальне/вертикальне відображення, поворот, вирізання участку зображення, штучне зашумлення даних та набагато складніші, як на рис (2.6). Крім збільшення датасету, це збільшує робастність моделі. Таким чином ми ще можемо збалансувати датасет по класам.

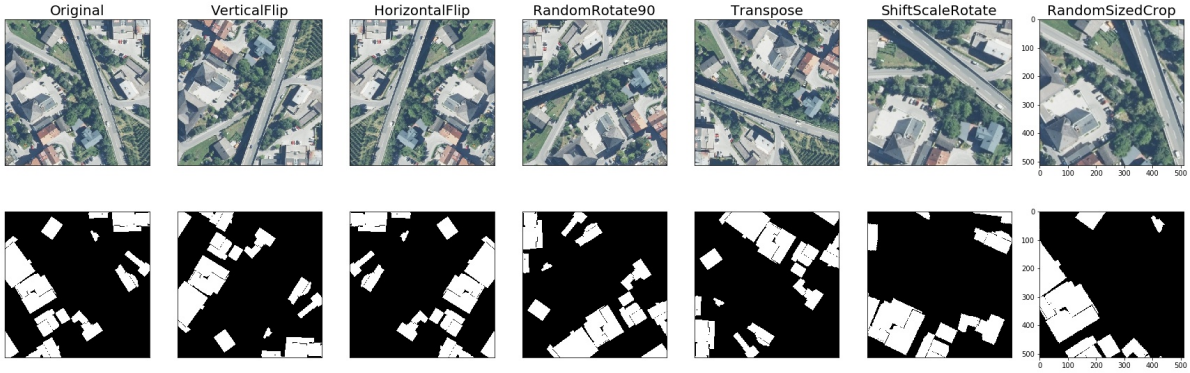


Рисунок 2.6

Аугментацію можна використовувати для різних задач, але в умовах класифікації потрібно бути акуратним. В рамках задачі розпізнавання рукописних цифр MNIST аугментація погіршує точність. На рис (2.7) немає цифри шість(6). Також треба бути акуратним з вирізанням участку - якщо на зображенні є інші об’єкти або наш цільовий об'єкт маленького розміру - є шанс що він не попаде в участок і модель буде лише неправильно вчитись.

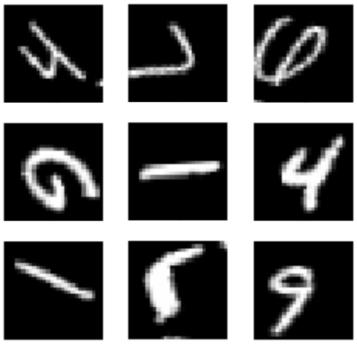


Рисунок 2.7

Експеримент проведений Jason Wang та Luis Perez у 2017 порівняння найрізноманітніших перетворень. Як видно на рис (2.8) модель без аугментації швидше вивчає навчальну вибірку і дає гірші результати на тесті. [13]

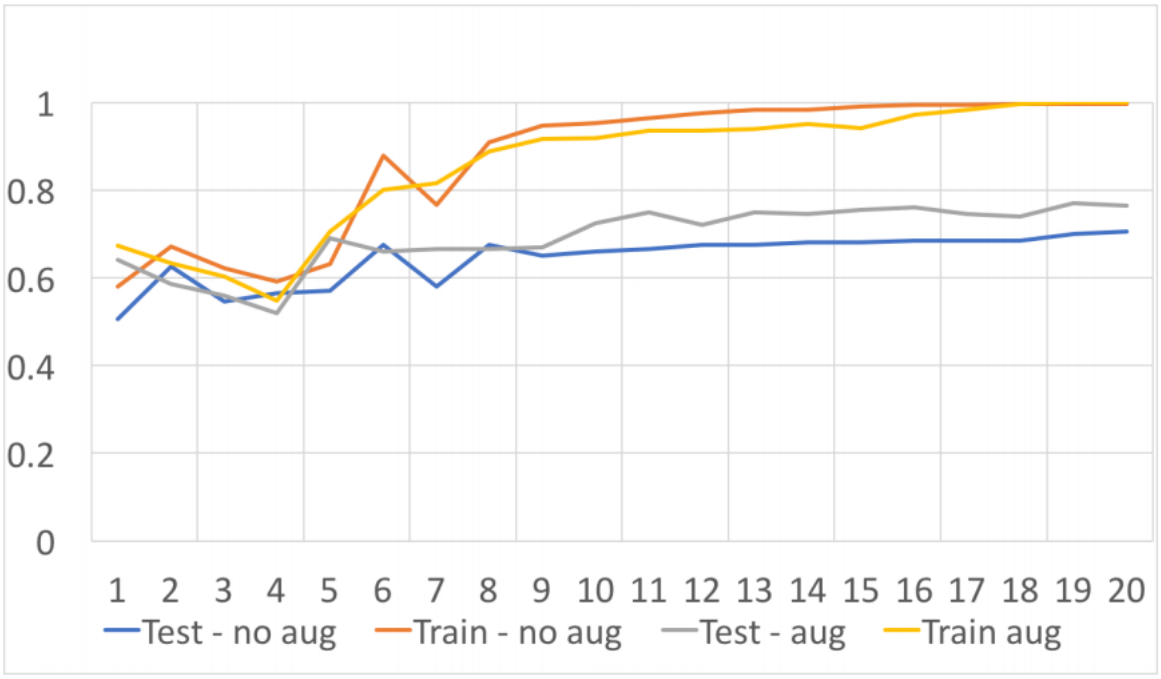


Рисунок 2.8

Ще одна техніка використання аугментації при прогнозуванні - Test Time Augmentation. Замість одного зображення подавати оригінальне + всі його аугментації. Для кожного перетворення отримати ймовірність належності до певного класу і взяти середнє по цим результатам.

2.7 Навчання на одному/кількох представників на класс

Це особливий випадок, коли навчальних даних один/кілька представників на кожен клас. В задачах Face Recognition намагаються знайти схожість (відстань) між двома лицями. Це зумовлено кількома особливостями:

1. Кількість людей величезна, і поява хоча б одного учасника змінювала розподіл ймовірностей належності лиця відповідній людині;

2. Лице - надзвичайно складний об'єкт дослідження. І 1 навчальний приклад не можу бути вивчений градієнтним спуском.

Але у випадку передавального навчання, ми не вивчаємо нейронну мережу з нуля. Якщо ми розглядаємо випадок AnB - ми зменшуємо простір пошук ознак, особливо якщо n-тий це класифікатор. Якщо це випадок AnB+, то ми лише робимо точну настройку під наше завдання. Застосування цього підходу разом з аугментаціями навчальних даних робить можливим навчання градієнтного спуску у випадку Few Shot learning.

2.8 Постановка задачі класифікації

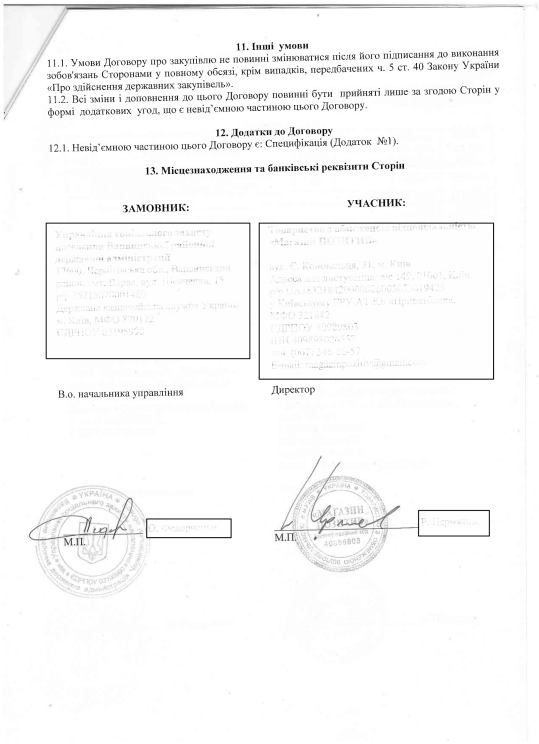
За вхідним зображення знайти до якого класу належить зображення. Множина класів має бути скінченна і завчасно задана. Так як останній шар нейронних мереж є SoftMax - він визначає ймовірність належності відповідному класу. Сумарна ймовірність повинна бути рівна 1. Якщо у нас з'явиться новий клас - необхідно перевчити класифікатор, а краще всю нейронну мережу. Для класифікації в рамках ImageNet використовують дві метрики top1 та top5 accuracity. top1 - це випадок звичайної точності формула (2.1). Серед масиву ймовірностей обирають найбільшу - це і є наша відповідь.

$$Acc = \frac{N_{correct}}{N_{All}} \tag{2.1}$$

top_n - це чи серед n найбільших ймовірностей присутній правильний клас. Тобто top5>=top1. Інколи на зображенні може бути кілька об'єктів різних класів. В моєму випадку такого бути не може, тому Acc - основна метрика для моїх моделей.

2.9 Набір даних

Наприклад на сайті Prozorro величезна кількість документів у вільному доступі. Відповідно до чинного українського законодавства та правил укладень договорів якщо на документі немає печатки зі сторони учасника або замовника, як на рис (2.9) - договір анулюється. Тому необхідно перевіряти на наявність печаток на документі всіх сторін.



11. Інші умови

11.1. Умови Договору про закупівлю не повинні змінюватися після його підписання до виконання зобов'язань Сторонами у повному обсязі, крім випадків, передбачених ч. 5 ст. 40 Закону України «Про здійснення державних закупівель».

11.2. Всі зміни і доповнення до цього Договору повинні бути прийняті лише за згодою Сторін у формі додаткових угод, що є невід'ємною частиною цього Договору.

12. Додатки до Договору

12.1. Невід'ємною частиною цього Договору є: Специфікація (Додаток №1).

13. Місцезнаходження та банківські реквізити Сторін

ЗАМОВНИК: **УЧАСНИК:**

В.о. начальника управління Директор

М.П. М.П.

Рисунок 2.9 Приклад договору з печатками двох сторін.

Хоча інформація є у відкритому доступі - більшість особистої інформації буде зашумлена. Підготовкою розмічених даних я займався самостійно. За допомогою api від Prozorro скачав 500 випадкових договорів. Кожен з них

містить різну кількість сторінок, тому сумарна кількість - 2045 сторінок. Кожна з них містить від 0 до 4 печаток. На рис (2.10) та на рис (2.11) представники кожного класу

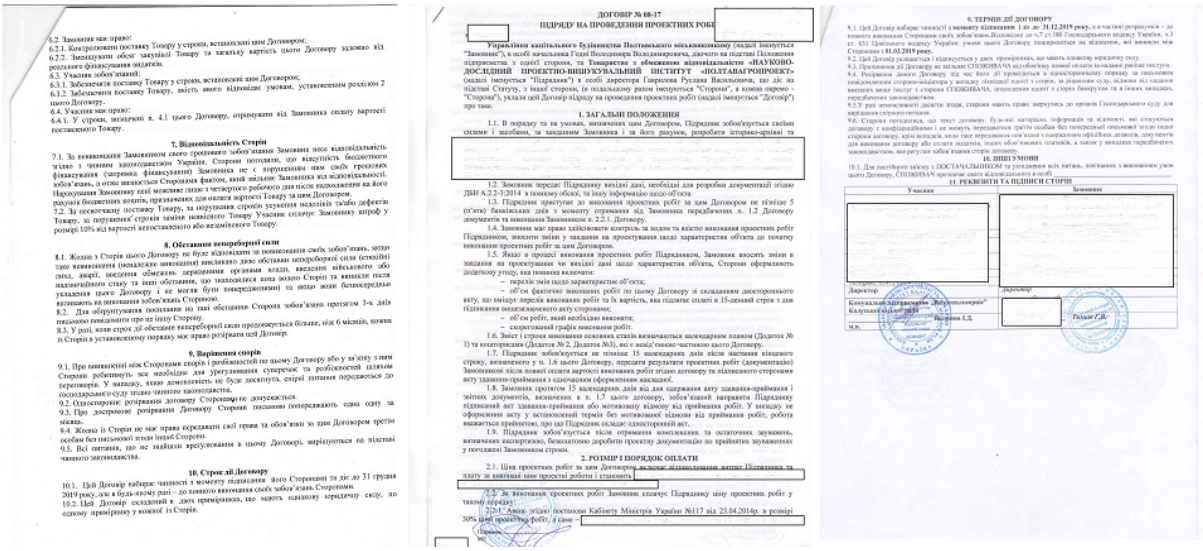


Рисунок 2.10

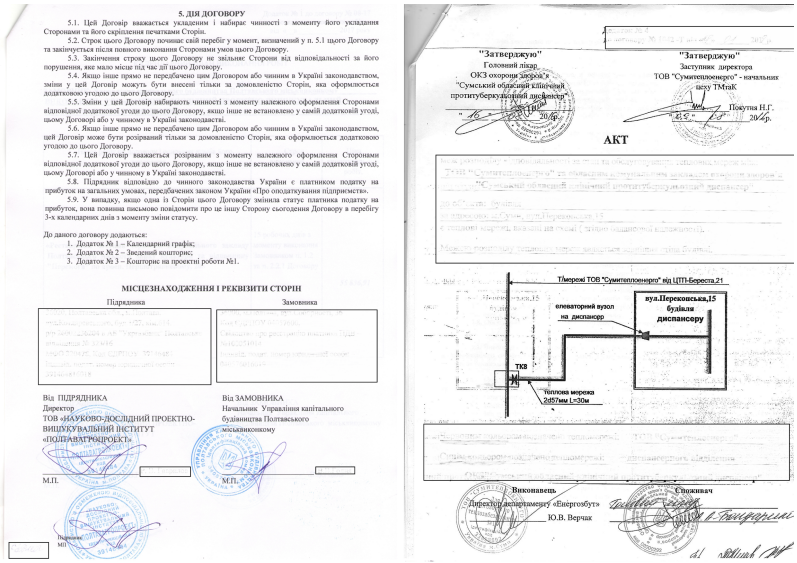


Рисунок 2.11

Частина зображень - кольорові скан-копії, чорно-білі скан-копії. Деякі це фотографії документів.

Частина документів перевернуті, ще частина зашумлена як на рис (2.12)

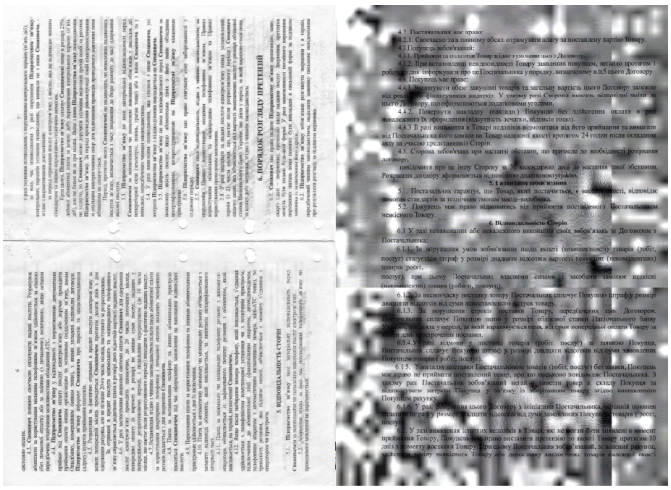


Рисунок 2.12

В більшості випадків висота документа в межах 3000-4000. Ширина в свою чергу майже завжди в районі 2400, як видно на рис (2.13)

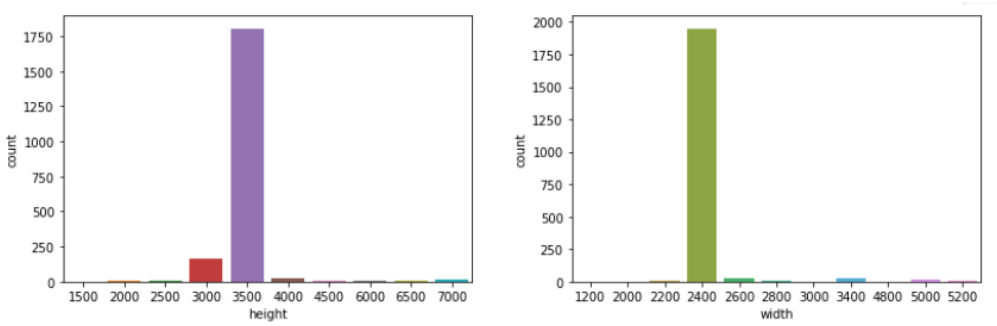


Рисунок 2.13

Розподіл цільового класу у табл (2.1)

Таблиця 2.1 - Розподіл цільових класів

Кількість печаток	Кількість прикладів
0	1135
1	121
2	760
3	27

4	2

2.10 Висновки до розділу 2

В даному розділі спершу було розглянути поняття завдання та доменної області. У випадку передавального навчання розглянули підходи передачі архітектури та коефіцієнтів шарів: з заморожуванням коефіцієнтів або їх точною настройкою.

Було розглянути поняття загального і унікального фільтру. На прикладі архітектури AlexNet в задачі класифікації ImageNet було розглянутий перехід від загальних до унікальних.

Також розглянутий підхід виділення ознак з подальшим використанням вектору в класичних підходах машинного навчання.

Розглянули підхід аугментації даних для збільшення датасету та робасності моделі. Та коли цей підхід буде лише погіршувати навчання.

Розглянули ситуацію one/few shot learning та чому його можливо використати з зміною коефіцієнтів при градієнтному спуску.

Зроблена постановка задачі класифікації. В рамках нашої задачі було визначено на які класи ми будемо розподіляти наші зображення. Зроблено дослідницький аналіз датасету.

РОЗДІЛ 3 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Вступ

В даному розділі будемо розглядати різні підходи до класифікації документів за кількістю на них печаток. Основна мета - дізнатись яка базова архітектура краща для передавального навчання. Та детально розглянути техніки заморожування коефіцієнтів для навчання і їх точна настройка. Дізнатись ефективність використання лінійної регресії та градієнтного бустингу над ембедінгами створеними різними архітектурами.

3.2 Вимоги та інструменти для обробки даних та навчання нейронної мережі

Навчання нейронної мережі та обробки даних відбувається за допомогою таких умов та інструментів:

- мова програмування Python 3.6
- Середовище Jupyter Notebook з використанням хмарної системи Kaggle; Kaggle дозволяє взаємодіяти з файлами .ipynb. Розробка моделей була на Kaggle Notebook. Взаємодія з даними була за допомогою системи Kaggle Dataset;
- взаємодія з Kaggle була за допомогою веб-браузера Google Chrome версії 83.

Kaggle безкоштовно надає потужності, але є квоти на використання графічного редактору 30 год в тиждень. Так як підхід передавального навчання має значно зменшити необхідні потужності та час на навчання NN тому всі експерименти можна проводити на комп'ютері з характеристиками:

- процесор Intel Pentium ;
- мова програмування Python 3.6;
- середовище Jupyter Notebook;
- операційна система не грає ролі;
- 7 гб на жорсткому диску для зберігання зображень та інформації про них.

Бібліотеки:

- numpy: робота з масивами та математичні операції;
- pandas: інструмент зчитування таблиць у форматі .csv та зручна взаємодія з ними;
- matplotlib: бібліотека для побудови графіків;
- sklearn: розбиття набору даних на навчальну та тестову вибірку; метрики точності; логістична регресія;
- cv2: для зчитування та обробки зображень;
- os: взаємодія з операційною системою;
- tensorflow: фреймворк для навчання, прогнозування NN. функція втрат, метрика для моніторингу під час навчання. Шари функції. Скачування архітектури та вагових коефіцієнтів;
- keras: надбудова над tensorflow, Theano, Deeplearning4j для зручної та простої взаємодіє з ними. підтримка всередині tensorflow;
- LightGBM: градієнтний бустинг LGBM.

3.3 Критерії оцінки моделей

Критерій порівняння - Точність описано в 2 розділі. Функція втрат - categorical_crossentropy. Для перевірки точності моделі та перенавчання ноної нейронної мережі розібємо наш датасет на 2 вибірки: train і valid. Так як навчальних даних мало (2 тис) проблема перенавчання може відбуватись надзвичайно швидко. При високому LR = 10^-2 може вивчити всю навчальну вибірку за 1 епоху. Показувати на train 100% точності, але буде абсолютно не здатна до узагальнення.

Так як в ході навчання буде 2 тис зображень, добавляти 3-ій тип test для фінальної вибірки не є доцільним. Також підбір гіпермараметрів відбувається лише за точністю і функції втрат на train; рання зупинка лише за train; розбиття відбувається під час кожного запуску - ми можемо вважати точністю на valid - об'єктивною. Розбиття буде 50% train і 50% valid.

Так як початкова задача про розпізнавання кількості печаток в рамках договору - не можна допустити, що частина документу в train, а частина в test. Але в той же час необхідно і не забувати про баланс класів. Так як в більшості документів сумарна кількість печаток або 2 або 4 - будемо розбивати рівномірно за сумою кількості печаток на документі. Документи у яких кількості печаток унікальна - надамо значення суми 100 і ці документи випадковим чином додаються до однієї з вибірок.

Для того щоб запобігти перенавчанню нейронної мережі використовують ранню зупинку. Це означає, якщо критерій не зменшується або метрика не покращується - ми припиняємо навчання та повертаємо вагові коефіцієнти до найкращих. Параметр EARLY_STOPPING відповідає за кількість епох які ми будемо чекати покращень.

Умови запуску будуть:

- dim = (224,224);
- EPOCHS = 10;
- batch_size = 64;
- n_classes = 5;
- n_channels = 3;
- EARLY_STOPPING = 3.

Якщо модель зустріла ранню зупинку або точність поступово виходила на плато - навчання закінчується. В іншому випадку ми продовжуємо навчати NN стільки епох, скільки необхідно. Метод навчання Adam(learning_rate=1e-3), якщо не вказаний інший.

3.4 CNN без передавального навчання

Спробуємо навчати нейронну мережу з випадково ініціалізованих вагів. Розглянуто стандартну архітектуру CNN з/без регуляризації на рис (3.1)

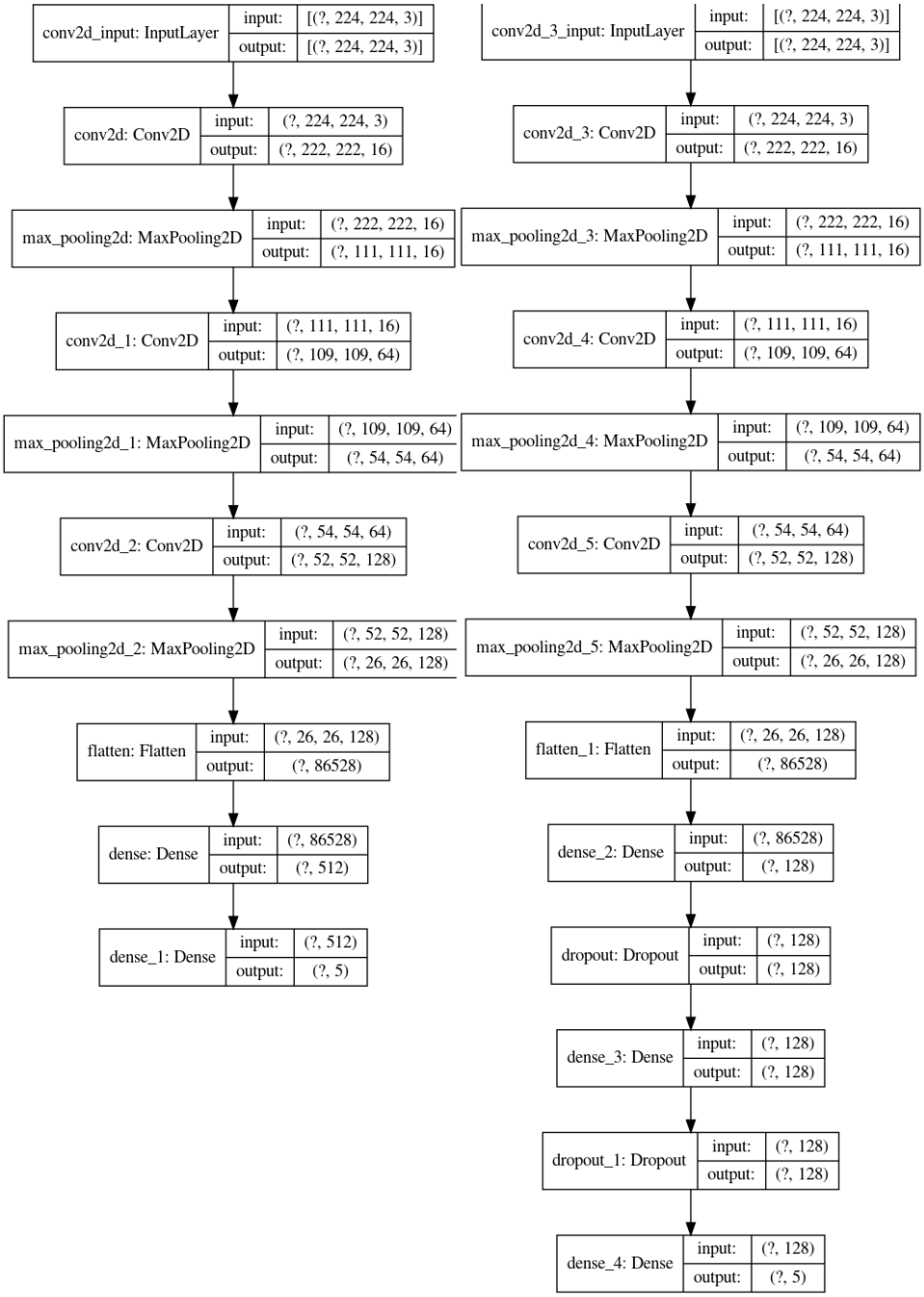


Рисунок 3.1 Архітектура CNN з та без регуляризації

Результати навчання CNN без регуляризації на рис (3.2):

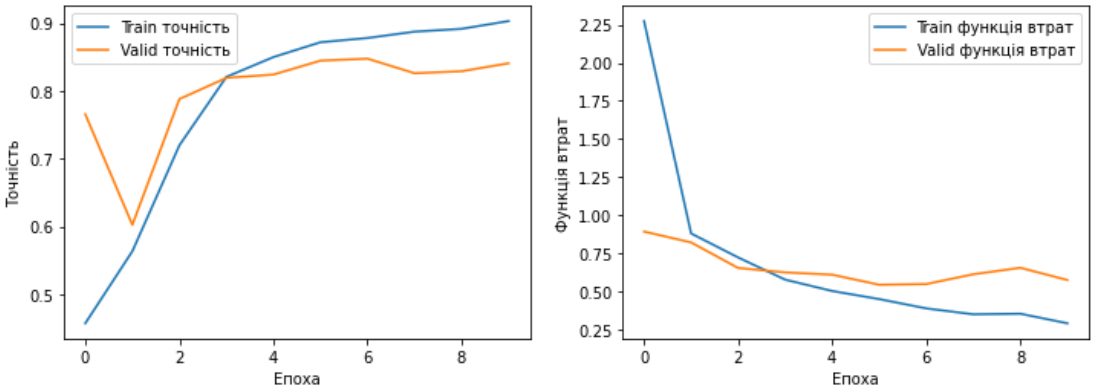


Рисунок 3.2

Найкраща точність на валідації 0.84765625 на 7 епосі

Результати навчання CNN з регуляризацією на рис (3.3):

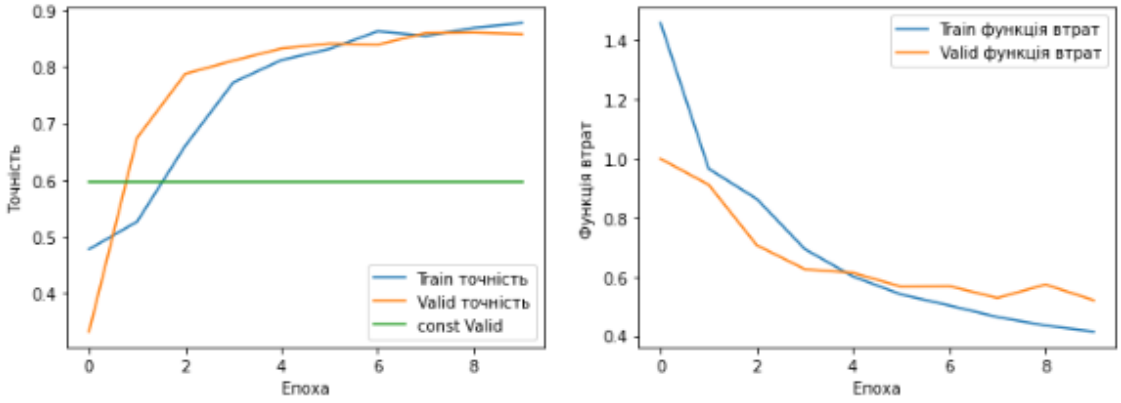


Рисунок 3.3

Найкраща точність на валідації 0.8613281 на 7 епосі.

Тепер розглянемо навчання CNN з регуляризацією та з аугментація на рис (3.4) . Навчання було на 20 епохах.

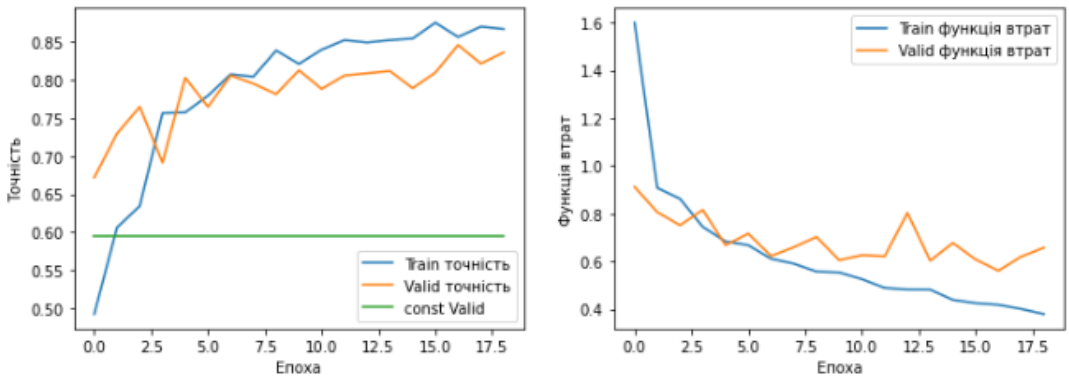


Рисунок 3.4

Найкраща точність 0.8457031 на 17 епосі.

Можемо зробити висновок що при навчанні CNN з випадковими коефіцієнтами навіть з маленьким датасетом регуляризація збільшує точність на 0.01367. А аугментація погіршує на 0.0156.

3.5 Log Reg and LGBM on embedding

Так як ми розглядаємо вже претренороване моделі - ми можемо використати їх для виділення ознак. В такому випадку ми беремо вхід, середину NN. Але замість того щоб на основі цих ознак прогнозувати багатошаровим персептроном ми обріжемо хвіст NN, як на рис (3.5). Далі будемо взаємодіяти не з зображенням напряму розмірності (224,224,3) а у випадку з VVG16 вектором розмірністю 25088. А на основі цього вектора робимо класифікацію класичними методами машинного навчання, як логістична регресія та градієнтний бустинг на деревах. значною перевагою цього метода - взагалі не потрібно навчати нейронну мережу. Але так як немає навчання - немає точної настройки шарів виділення ознак. Якщо у випадку логістичної регресії - це аналог MLP, з вбудованою регуляризацією, то у випадку дерев ситуація складніше. Так як вектор розмірністю 25088, а дерева за один поділ вершини можуть перевірити лише одну ознаку. Також цей підхід сприяє перенавчанню на навчальній вибірці,

бо звертає увагу лише на кілька ознак, але з деякими архітектурами показує чудовий результат навіть без коригування гіперпараметрів.

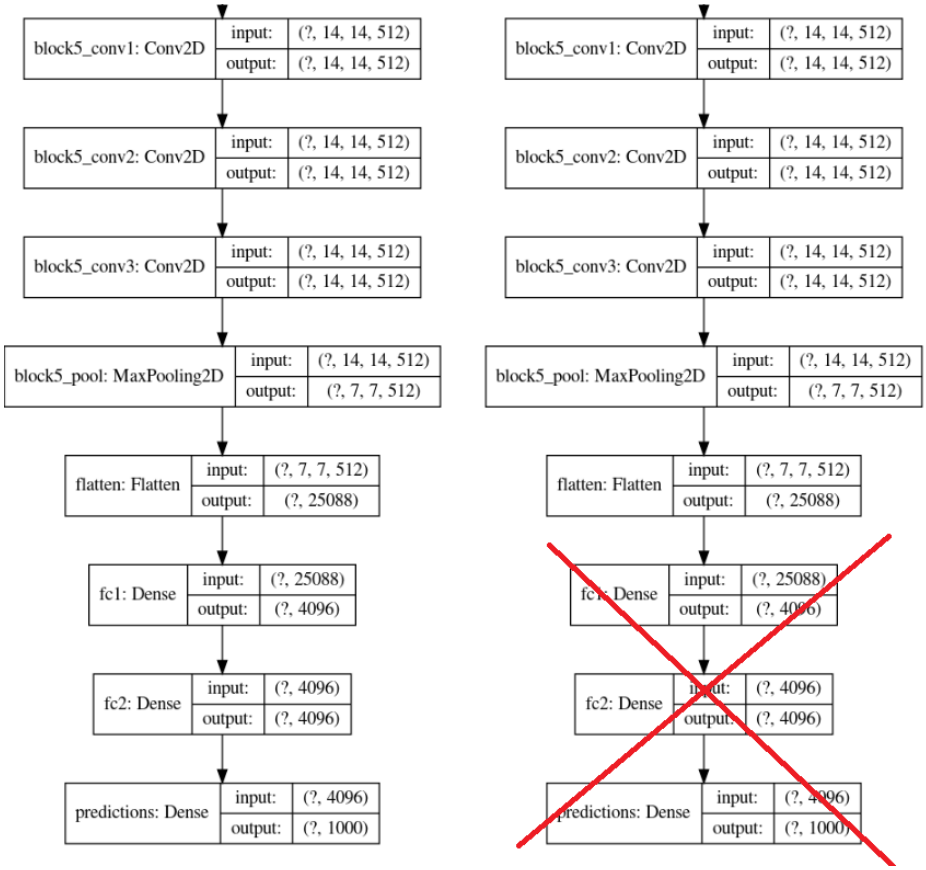


Рисунок 3.5 - Приклад обрізання класифікатора

Для кожної архітектури необхідно визначити, де закінчується виділення ознак, а де починається класифікація.

Розмірність вхідних даних: так як вагові коефіцієнти були отримані при навчанні на (224,224,3) необхідно змінити розмірність наших зображень. Цей параметр необхідно дивитись в документації кожної моделі.

У випадку VGG16/19 класифікатор складається з 3 FC шарів. Так як навчання нейронної мережі немає - краще не поєднувати цих два підходи: на вхід логістичної регресії подавати вихід першого FC шару наслід, як видно з табл (3.1).

Таблиця 3.1

VGG16 output	FC1	Flatten
LogReg	0.8958	0.9045
LGBM	0.8823	0.8958

Для всіх архітектур було застосовані стандартні гіперпараметри.

Найкращі вийшли архітектури VGG16 та VGG19 та MobileNet, як видно з табл (3.2). Так як VGG16 та VGG19 практично однакові за архітектурою - кращу 16-у. Не будемо забувати, що одна з найбільший переваг MobileNet - швидкодія.

Таблиця 3.2

Арітектрура/ Модель	LGBM	LogReg	Фінальна модель
VGG16	0.8958	0.9045	0.9045
VGG19	0.8948	0.8977	0.8977
ResNet50	0.8813	0.8862	0.8862
InceptionV3	0.7984	0.7512	0.7984
DenseNet121	0.8244	0.7675	0.8244
MobileNet	0.8804	0.8929	0.8929
MobileNetV2	0.8273	0.8322	0.8322
Найкращий по методу	0.8958	0.9045	0.9045

3.6 Замороження для навчання та точна настройка

Тут ми розглянемо ефективність замороження для навчання та точна настройки. Експеримент будемо проводити в однакових умовах для всіх випадків. А саме до ранньої зупинки - якщо точність не буде зростати 5 кроків -

ми зупиняємо навчання і беремо найкращий результат на валідації. Архітектура VGG16; класифікатор на рис (3.6).

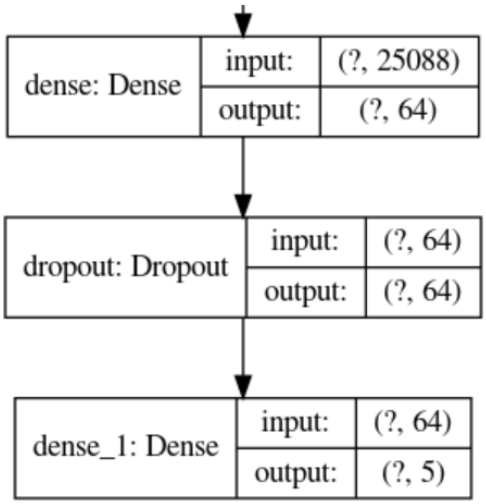


Рисунок 3.6

Так як даних дуже мало - не будемо використовувати складний класифікатор.

Як можемо бачити на рис (3.7), що перших кілька шарів точність не зменшується, а і покращується, якщо будемо заморожувати 5 шарів. Це можна пояснити, що вони загальні і їхнє навчання необов'язкове. Адже вони були навчені на набагато більгому наборі даних і краще здатні до узагальнення. Якщо заморожувати ще до 10 шарів точність поступово зменшується. Тут починають діяти проблеми описані в 2 розділі, а саме:

1. Перехід від загальних шарів до унікальних;
2. Набагато сильніше починає впливати різниця датасетів.

Після заморожування більгості шарів і точність стрімко погіршується.

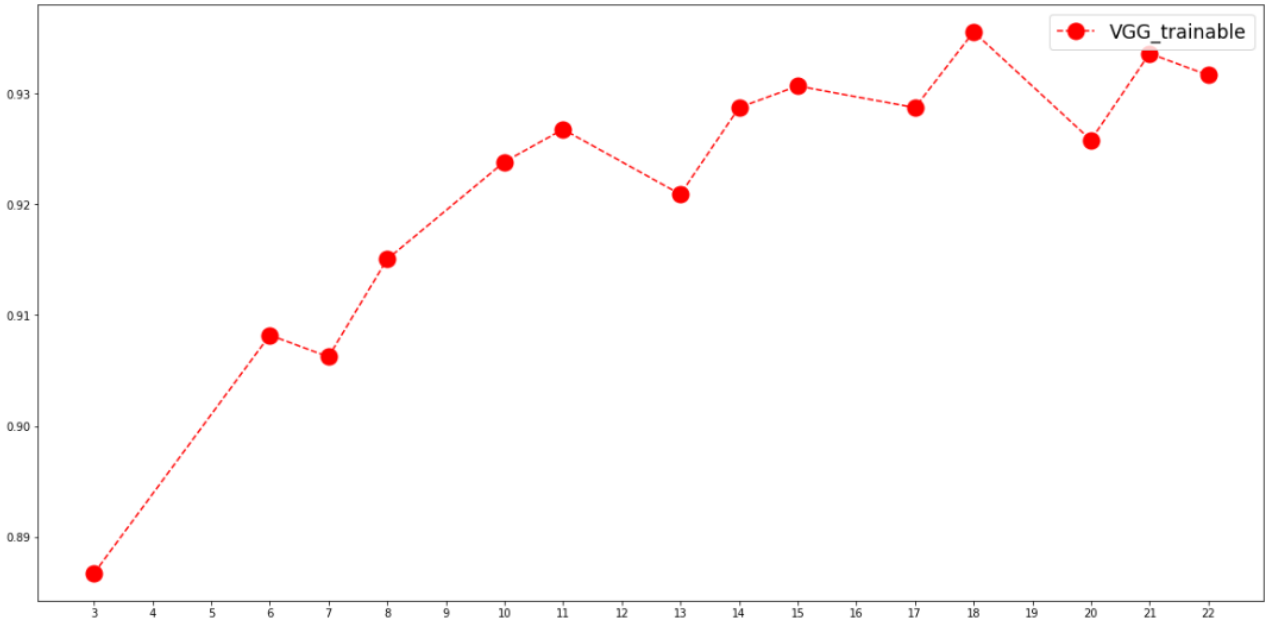


Рисунок 3.7 - залежність точності від кількості розморожених шарів для навчання

3.6 One/Few Shot Learning

Яка буде точність нейронної мережі, якщо використовувати навчання але у випадку коли даних були надзвичайно мало - 25 елементів можемо побачити на рис (3.8). Для збільшення розміру навчальної вибірки будемо використовувати аугментацію:

Дзеркальне відображення відносно вертикалі і горизонталі з шансом 10% при навчанні нейронної мережі.

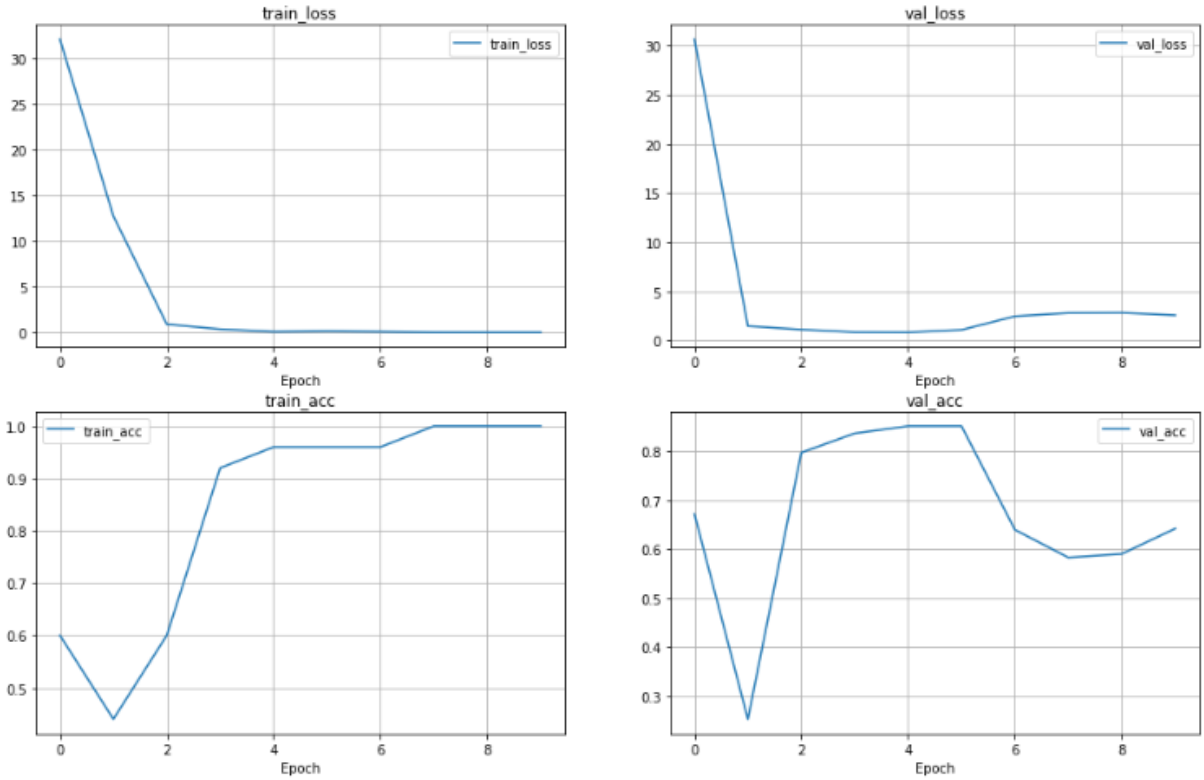


Рисунок 3.8

Умови це - Train - 25 , Validation - 405. Найкраща точність - 0.8558519 на 5 епосі. Learning rate у цьому випадку був 1e-2.

3.7 Висновок до розділу 3

Можемо зробити висновок з рис (3.9), що використання передавального навчання ефективно у порівнянні з звичайним підхом. У випадку архітектури VGG16 ми можемо використовувати перших 12 заморожених шарів. Якщо використовувати більше - вже краще використати нейронну мережу для виділення ознак. Як бачимо, навіть якщо навчальна вибірка у 40 разів менше у порівнянні з звичайним підходом навчання NN - передавальне показує приблизну таку саму точність.

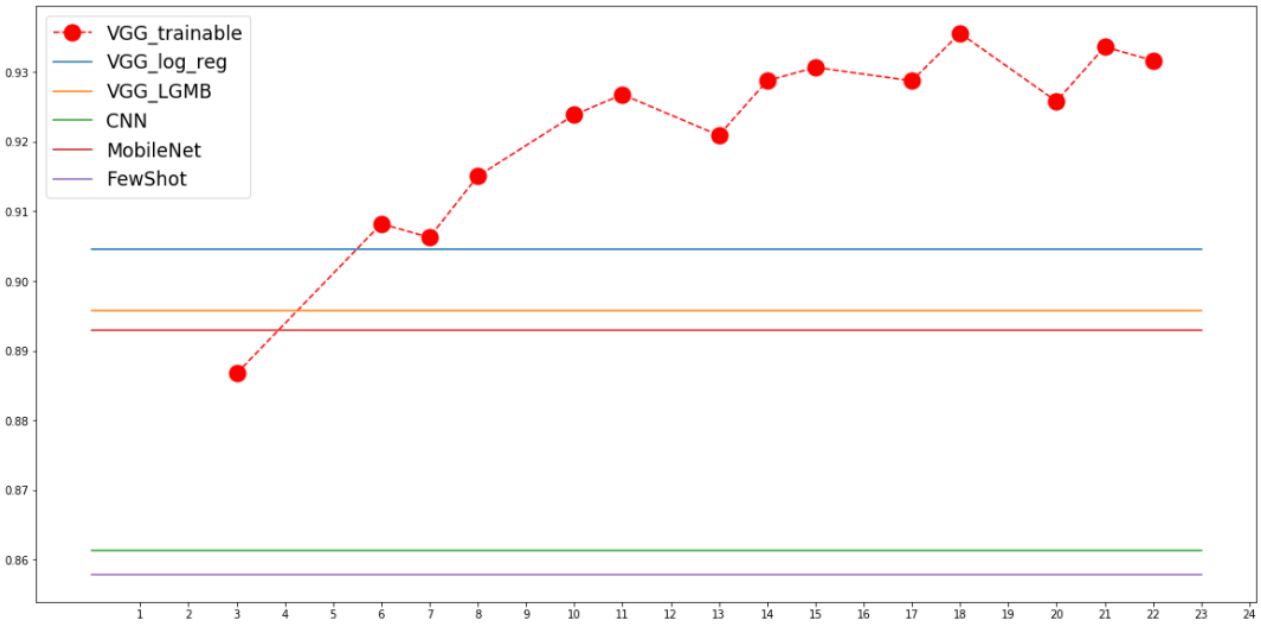

VGG_trainable
VGG_log_reg
VGG_LGMB
CNN
MobileNet
FewShot
0.93
0.92
0.91
0.90
0.89
0.88
0.87
0.86
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

Рисунок 3.9

РОЗДІЛ 4 ФУНКЦІОНАЛЬНИЙ-ВАРТІСНИЙ АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ У ОБЛАСТІ КОМП'ЮТЕРНОГО ЗОРУ

4.1 Постановка задачі

Проводиться оцінка основних характеристик результатів та їх собівартість. Для отримання результатів використовувалась мова Python. Середовище розробки - Jupyter Notebook. Робота програми не залежить від технологій реалізації апаратного забезпечення та операційної системи. Нижче приведено аналіз різних підходів до навчання нейронної мережі для задачі розпізнавання і класифікацій особливостей документів.

4.2 Обґрунтування функцій та параметрів дослідження

Основні функції:

F_1 - кількість розмічених даних; F_2 - вибір підходу до розв'язку задачі комп'ютерного зору; F_3 - вибір мови програмування

ФункціяF_1: а) 100 розмічених документів; б) 2000 розмічених документів; в) 50000 розмічених документів

Функція F_2: а) Передавальне навчання для навчання нейронної мережі; б) Звичайний підхід комп'ютерного зору для навчання нейронної мережі

Функція F_3 : а) Вибір мови програмування Python; б) Вибір мови програмування C++; в) Вибір мови програмування Matlab;

Знизу на рис (4.1) відповідну морфологічну карту системи:

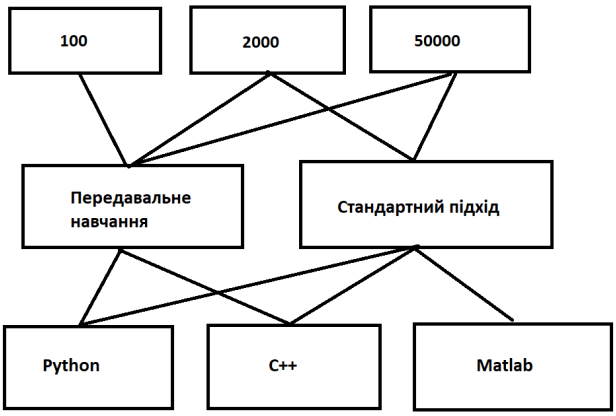


Рисунок 4.1 - Морфологічна карта

Позитивно-негативна матриця варіантів основних функцій (табл. 4.1):

Таблиця 4.1

Основні функції	Варіанти реалізацій	Переваги	Недоліки
F_1	А	Надзвичайно швидко отримати. Не обов'язкове спеціальне програмне забезпечення. Не займає багато місця на комп'ютері	Нейронній мережі може бути замало даних для навчання
F_1	Б	Отримати ці дані буде не дуже довго. нейронній мережі може вистачити даних. 6 Гб на комютері	Необхідне програмне забезпечення
F_2	В	Достатня кількість даних для нейронної мережі	Необхідне програмне забезпечення. Величезна кількість часу для отримання даних. Десь треба зберігати 150 Гб
F_1	А	Не потрібні потужні характеристики компьютеру. Навчання проходить надзвичайно швидко	Обмеженість у виборі архітектури нейронної мережі

F_2	Б	Гнучкість відносно архітектури	Потрібні потужні характеристики компьютеру. Навчання проходить надзвичайно довго
F_2	А	Вже готові бібліотеки та алгоритми. Ідеально підходить для передавального навчання.	Менша швидкість
F_3	Б	Висока швидкість	Створення спеціальних алгоритмів навчання є надзвичайно складне та довге завдання. Власний алгоритм може виявитись повільніший ніж на Python
F_3	В	Зручність для користувача	Несумісність з передавальним навчання. Необхідне програмне забезпечення ціною в 1000$/ рік для кожного розробника.

Тепер, за наявності позитивно-негативної матриці можна робити висновки щодо доцільності використання одних варіантів та недоцільності використання інших:

На основі порівняльного аналізу варіантів реалізації основних функцій по їх перевагам і недолікам можна виключити варіанти F_1 а і в , F_3 б і в, тоді варіанти, які залишилися:

F_1 б) - F_2 а) - F_3 а)

F_1 б) - F_2 б) - F_3 а)

Для оцінювання описаних функцій запропонована система параметрів. Опишемо цю систему.

4.3 Обґрунтування системи параметрів досліджень

Для характеризації досліджень пропонуємо такі параметри:

X1 — точність моделі на небачених даних (% неправильно класифікованих документів); X2 - навчання нейронної мережі (час, навчання нейронної мережі); X3 - складність освоєння теоретичної бази(час на освоєння теоретичної бази); X4 — Час на освоєння мови програмування (Час на вивчення мови програмування); X5 - складність освоєння бібліотеки tensorflow(Час на освоєння бібліотеки tensorflow); X6 — Оперативна память (кількість оперативної пам’яті для роботи).

Взаємозв'язок F_1 - X6; F_2 -X1,X2; F_3 - X3, X4, X5;

Гірші, середні та кращі показники параметрів вибираються на основі вимог замовника та умов перебігу дослідження, їх наведено у таблиці (4.2) та їх графічне зображення на рис (4.2)

Таблиця 4.2 — основні параметри дослідження

Умовні позначення	Одиниці виміру	Гірші	Середні	Кращі
X1	%	30	10	4
X2	Год.	50	5	2
X3	Год.	30	20	15
X4	Год.	20	12	5
X5	Год.	20	10	6
X6	Гб.	8	12	16

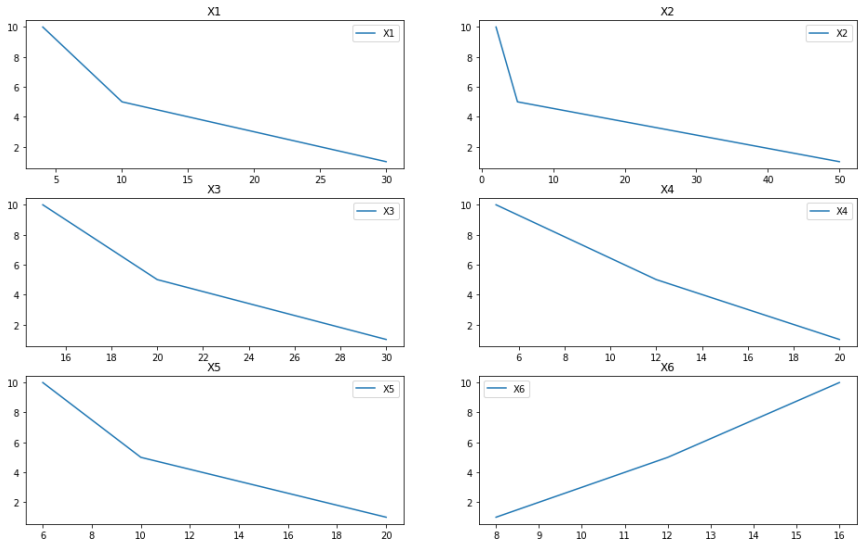


Рисунок 4.2 - Графічні характеристики описаних вище параметрів

Вагомість параметрів визначається методом попарного їх порівняння на основі результатів ранжування експертами та попарного порівняння параметрів. Наведемо результати експертного ранжування(таблиця 4.3): — достовірне, виходячи із даної нерівності. Параметр 1 - найважливіший; 6 - найменш.

Таблиця 4.3 — результати експертного ражування параметрів.

Умовне позначення	Одиниці вимірювання	Ранг експерта1	2	3	4	5	6	7	Сума рангів, R_i	Відхилення, Δ_i	Δ^2_i
X1	Год.	1	2	1	1	1	3	1	10	-14.5	210.2
X2	Год.	3	1	2	3	3	1	2	21	-9.5	90.25
X3	Год.	2	3	3	2	2	2	3	20	-7.5	6.25
X4	Год.	6	6	5	6	6	6	4	39	14.5	10.25
X5	Год.	5	4	6	4	5	4	5	26	8.5	2.25
X6	Гб.	4	5	4	5	4	5	6	31	8.5	2.25
Разом		21	1	1	1	1	1	1	147	0	711.5

За даними табл. 4.3 визначимо коефіцієнт конкордації за формулою:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12*711.5}{7^2(6^3 - 6)} \approx 0.8297 \ > \ 0.67$$

Таблиця 4.4

Параметри	1	2	3	4	5	6	7	Кінцева оцінк	Числове значення
X1 та X2	<	>	<	<	<	>	<	<	1.5
X1 та X3	<	<	<	<	<	>	<	<	1.5
X1 та X4	<	<	<	<	<	<	<	<	1.5
X1 та X5	<	<	<	<	<	<	<	<	1.5
X1 та X6	<	<	<	<	<	<	<	<	1.5
X2 та X3	>	<	<	>	>	<	<	<	1.5
X2 та X4	<	<	<	<	<	<	<	<	1.5

X2 та X5	<	<	<	<	<	<	<	<	1.5
X2 та X6	<	<	<	<	<	<	<	<	1.5
X3 та X4	<	<	<	<	<	<	<	<	1.5
X3 та X5	<	<	<	<	<	<	<	<	1.5
X3 та X6	<	<	<	<	<	<	<	<	1.5
X4 та X5	>	>	<	>	>	>	<	>	0.5
X4 та X6	>	>	>	>	>	>	<	>	0.5
X5 та X6	>	<	>	<	>	<	<	<	1.5

4.4 Аналіз рівня якості варіантів реалізації функцій

Розрахунок вагомості занотуємо у таблицю 4.5, показники рівня якості у таблицю 4.6:

Таблиця 4.5

Xi/XJ	1	2	3	4	5	6	$b^1{}_i$	$K^1{}_i$	$b^2{}_i$	$K^2{}_i$	$b^3{}_i$	$K^3{}_i$
X1	1	1.5	1.5	1.5	1.5	1.5	8.5	0.236	49.75	0.250	272.87	0.251
X2	0.5	1	1.5	1.5	1.5	1.5	7.5	0.208	41.75	0.210	227.1	0.209
X3	0.5	0.5	1	1.5	1.5	1.5	6.5	0.180	34.75	0.175	188.87	0.173
X4	0.5	0.5	0.5	1	0.5	0.5	3.5	0.097	19.75	0.099	109.12	0.100
X5	0.5	0.5	0.5	1.5	1	1.5	5.5	0.152	28.75	0.144	157.12	0.144
X6	0.5	0.5	0.5	1.5	0.5	1	4.5	0.12	23.75	0.119	130.87	0.120

Таблиця 4.6

Основні функції	Варіанти реалізацій	Параметри	Абсолютне Значення	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	б)	X6	12	5	0.120	0.6
F_2	а)	X1	6	8.33	0.251	2.09
F_2	а)	X2	2	10	0.209	2.09
F_2	б)	X1	15	4	0.251	1.004
F_2	б)	X2	3	8.33	0.209	1.74
F_3	а)	X3	18	7	0.173	1.21
F_3	а)	X4	10	6.33	0.100	0.633
F_3	а)	X5	10	5	0.144	0.72

K_1 = 0.6+ 2.09 + 2.09 +1.21 +0.633+0.72 = 7.34

K_2 = 0.6+ 1.004 + 1.74 +1.21 +0.633+0.72 = 5.9

Отже, перший варіант, що передбачає передавальне навчання дає більший результат. Тому віддаємо йому перевагу.

4.5 Економічний аналіз варіантів розробки ПП

Обидва варіанти включають в себе три окремих етапи:

1. підготовки даних
2. навчання нейронної мережі
    а) Передавальним навчання
    б) Звичайним підходом
3. розробка програмного продукту

Для завдання 1 (Алгоритм складності 3, ступінь новизни Г, вид використаної інформації БД) T_p =12, K_n = 0.3, K_{CK} = 0.7, $K_{CT.M}$ = 1.2

T_1 = 12*0.3*0.7*1.2 = 3.024

Для завдання 2 (при реалізації варіанту 1)), (Алгоритм складності 1, ступінь новизни Б, вид використаної інформації ПІ) T_p =64, K_n = 2.02, K_{CK} = 0.8, $K_{CT.M}$ = 1.6

T^1_2 = 64* 2.02 * 0.8 * 1.6 = 165.478 людино-днів

Для завдання 2 (при реалізації варіанту 2)), (Алгоритм складності 1, ступінь новизни Б, вид використаної інформації ПІ) T_p =64, K_n = 2.02, K_{CK} = 0.8, $K_{CT.M}$.м = 1.4

T^2_2 = 64* 2.02 * 0.8 * 1.4 = 144.793 людино-днів

Для завдання 3(Алгоритм складності 1, ступінь новизни В, вид використаної інформації БД) T_p =43, K_n = 1.35, K_{CK} = 0.6, $K_{CT.M}$ = 1.5

T_3= 43*1.35*0.6*1.5 = 52.24 людино-днів
T_1 = (3.024 + 165.478 + 52.24) * 8 = 1765 людино-годин
T_2 = (3.024 + 144.79 + 52.24) * 8 = 1600 людино-годин

В розробці та проведенні дослідження приймають участь 2 програмісти з окладом 12 000 грн

Зарплата розробників становить погодинно:

$$C = \frac{12000 + 12000}{2*21*8} = 71.4 \text{ грн}$$

Зарплата поваріантно

C_1 = 1765*71.4 = 126087 грн
C_2 = 1600*71.4 = 114270 грн

Відрахування на соціальний внесок становить 22%:

$C^{v}_1 = 0.22 * 126087 = 27739$ грн
$C^{v}_2 = 0.22 * 114270 = 25139$ грн

Визначаємо витрати на оплату однієї машино-години. З урахуванням заробітної плати програміста в розмірі 12000 грн з коефіцієнтом зайнятості 0.2, маємо

$C_Г = 12*М*K_З = 12 *12000* 0,2 = 28800$ грн

З урахуванням додаткової заробітної плати: СЗП =СГ* (1+ КЗ) = 28800 * (1 + 0.2)=34560 грн

Відрахування на соціальний внесок: $C_{ВІД} = C_{ЗП} * 0.22 = 34560 * 0,22 = 7603$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн

$C_A = K_{ТМ}* K_A*Ц_{ПР} = 1.15 * 0.25 * 8000 = 2300$ грн.

Витрати на ремонт та профілактику можна підрахувати:

$C_P = K_{ТМ}*Ц_{ПР} * K_P = 1.15 * 8000 * 0.05 = 460$ грн.,

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$T_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) *t_3*K_В = (365 – 104 – 11 – 16) * 8 * 0.9 = 1684.8$

Тепер рахуємо витрати на оплату електроенергії (з урахуванням ПДВ):

$C_{ЕЛ} = T_{ЕФ}*N_C* K_З*\ Ц_{ЕН}$=1684.8 * 0,3* 2 * 1,75 = 1769.46 грн.

Накладні витрати рахуються наступним чином:

$C_Н = Ц_{ПР}$*0.67 = 8000· 0,67 =5360 грн

Річні експлуатаційні витрати:

$C_{ЕКС}$ = 34560 + 7603 + 2300 + 460 + 1769.8 + 5360 = 52052 грн

Собівартість однієї машино-години ЕОМ становитиме:

$C_{М-Г} = C_{ЕКС}/ T_{ЕФ}$ = 52052/1684.8 = 30.9 грн/год

Витрати на оплату машинного часу складають в залежності від варіанту:

$C^1_М$ = 31.5 * 1765 = 55656 грн
$C^2_М$ = 31.5 * 1600 =50400 грн

Накладні витрати складають 67% від заробітної плати:

$C^1_Н$ = 0.67 * 126087 = 84478 грн
$C^2_Н$ = 0.67 * 114270 = 76560 грн

Таким чином, вартість розробки ПП та проведення дослідів складає:

C_1 = 126087 + 27739 + 55656 + 84478 = 292842 грн
C_2 = 114270 + 25139 + 50400+ 76560 = 265409 грн

Проведемо розрахунок техніко-економічного рівня:

$K_{ТЕР1}$ = 7.34 / 292842 = 2.507 * 10^-5
$K_{ТЕР2}$ = 5.9 / 265409 = 2.223 * 10^-5

Отже найбільш ефективним є перший варіант з коефіцієнтом техніко економічного рівня 2.507 * 10^-5.

За результатами проведеного функціонально-вартісного аналізу, можна зробити висновок, що з альтернатив, що залишилися після відбору, було отримано два варіанти.

Таким чином, після першого відбору було отримано два варіанти, в результаті проведення функціонально-вартісного аналізу стало зрозуміло, що доцільним є використання першого варіанту реалізації продукту. першому варіанту відповідає навчання нейронної мережі на 2000 документів з підходом передавального навчання на мові програмування Python. Таким чином, отримано нейронну мережу, яка з прийнятною точністю класифікує документи, не потребує великих об'ємів даних.

Висновок

В роботі було виконано дослідження ефективності використання передавального навчання для NN в задачі класифікації документів за кількістю печаток. Було проведений огляд найсучасніших архітектур для задач класифікації, які показали гарний результат на щорічному змаганні ImageNet. Було розглянуто передумови для використання технології передавального навчання. А також умови, що можуть завадити для покращення результату нейронної мережі, а саме: різниця між датасетом на якому навчалась нейронна мережа та цільовому датасеті; використання унікальних, а не загальних, шарів нейронної мережі.

Було зібрано та підготовлено маленький датасет скан-копії документів договорів з різної кількістю печаток.

Було досліджено використання претренерованих нейронних мереж для виділення ознак для подальшої роботи звичайними підходами машинного навчання. В результаті архітектура VGG16 показала найкращий результат, саме тому її було розглянути більш детально.

Під час дослідження ефективності нейронної мережі при різних кількостях заморожених шарів можемо зробити висновок, що перших 5 шарів - загальні і їх можна використовувати без великої втрати або навіть зростанням точності. Якщо використовувати 12 заморожених шарів - точність поступово починає спадати, але час навчання зменшується в 2-3 рази. Якщо заморожувати більшу половину шарів - точність починає стрімко падати і вже краще використовувати претренеровану мережу для виділення ознак і використання на них Логістичної регресії.

Фінальна точність звичайного підходу 86.1% точності, а для передавального 93.55%. Враховуючи, що приблизно 60% це документи з 0 печаток, то перевага стає ще більш вагомою.

Розглянуто випадок передавального навчання Few shot learning - коли навчальна вибірка це кілька представників на клас. І навіть коли вибірка 25

елементів, що в 40 разів менше від звичайного підходу - точність не набагато гірша.

Результатом роботи є нейронна мережа навчена на мінімальному об'ємі даних для класифікації документів за кількістю печаток з точністю 93.55%.

Подальші дослідження:

- Спробувати інші сучасні архітектури, наприклад MobileNet та ResNet50, для дослідження їх шарів на загальність.
- Провести аналогічний експеримент з навчальною вибіркою 20000+ елементів, щоб визначити наскільки сильно впливає відстань між датасетами;
- Знайти залежність між точністю і кількістю представників в кожному класі (Few Shot learning).

ПЕРЕЛІК ПОСИЛАНЬ

1. Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li Empirical Evaluation of Rectified Activations in Convolutional Network [Електронний ресурс] – URL: https://arxiv.org/abs/1505.00853 (дата звернення 12.05.2020)
2. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov; 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Електронний ресурс] – URL: http://jmlr.org/papers/v15/srivastava14a.html (дата звернення 12.05.2020)
3. Yann Lecun LeNet-5, convolutional neural networks [Електронний ресурс] – URL: http://yann.lecun.com/exdb/lenet/ (дата звернення 12.05.2020)
4. Karen Simonyan, Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition [Електронний ресурс] – URL: https://arxiv.org/abs/1409.1556 (дата звернення 12.05.2020)
5. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition [Електронний ресурс] – URL: https://arxiv.org/abs/1512.03385 (дата звернення 12.05.2020)
6. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going Deeper with Convolutions [Електронний ресурс] – URL: https://arxiv.org/abs/1409.4842 (дата звернення 12.05.2020)
7. Min Lin, Qiang Chen, Shuicheng Yan, Network In Network [Електронний ресурс] – URL https://arxiv.org/abs/1312.4400 (дата звернення 12.05.2020)
8. Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi, Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning [Електронний ресурс] – URL: https://arxiv.org/abs/1602.07261 (дата звернення 12.05.2020)
9. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Електронний

ресурс] – URL: https://arxiv.org/abs/1704.04861 (дата звернення 12.05.2020)

10. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, You Only Look Once: Unified, Real-Time Object Detection [Електронний ресурс] – URL: https://arxiv.org/abs/1506.02640 (дата звернення 12.05.2020)
11. Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson, How transferable are features in deep neural networks? [Електронний ресурс] – URL: https://arxiv.org/abs/1411.1792 (дата звернення 12.05.2020)
12. Simon Kornblith, Jonathon Shlens, Quoc V. Le, Do Better ImageNet Models Transfer Better? [Електронний ресурс] – URL: https://arxiv.org/abs/1805.08974 (дата звернення 12.05.2020)
13. The Effectiveness of Data Augmentation in Image Classification using Deep Learning [Електронний ресурс] – URL: https://arxiv.org/abs/1712.04621 (дата звернення 12.05.2020)

ДОДАТОК А

Програмний код для виконання тестових прикладів

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Input, Conv2D,MaxPooling2D, Flatten,
Dense,Dropout
from keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Model
from keras.utils import plot_model

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2

from sklearn.model_selection import train_test_split
from tqdm import tqdm
from sklearn.metrics import accuracy_score, confusion_matrix


    v_c_df = df_param['count_target'].value_counts().reset_index()
    one_represent = v_c_df.loc[v_c_df['count_target'] == 1, 'index'].tolist()
    df_param.loc[df_param['count_target'].isin(one_represent), 'count_target'] = 100
    return df_param

def df_split(df):

    count_target =
df.groupby(by=['tender']).target.sum().reset_index().rename(columns={'target':'count_
target'})
    count_target = create_one_represent_class(count_target)

    img_train, img_val  = train_test_split(count_target, test_size=0.2, random_state=42,
stratify=count_target['count_target'])

    train_df = df[df['tender'].isin(img_train['tender'])].reset_index(drop=True)
    test_df = df[df['tender'].isin(img_val['tender'])].reset_index(drop=True)

    return train_df, test_df

train_df, test_df  = df_split(df)
print(train_df.shape[0], test_df.shape[0])
```

```
class Data_generator(keras.utils.Sequence):
    def __init__(self, list_IDs, labels, batch_size=32, dim=(224,224), n_channels=3,
                 n_classes=5, shuffle=True):

        'Initialization'
        self.dim = dim
        self.batch_size = batch_size
        self.labels = labels
        self.list_IDs = list_IDs
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.shuffle = shuffle
        self.on_epoch_end()

    def on_epoch_end(self):
        'Updates indexes after each epoch'
        self.indexes = np.arange(len(self.list_IDs))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

    def __data_generation(self, list_IDs_temp):
        # Initialization
        X = np.empty((self.batch_size, *self.dim, self.n_channels))
        y = np.empty((self.batch_size), dtype=int)

        # Generate data

        for i, ID in enumerate(list_IDs_temp):
            X[i,] = (cv2.resize( cv2.imread(PATH_TRAIN+ID), dim ))

            # Store class
            #y[i] = self.labels[ID]
            y[i] = df[df.id == ID].target

        return X/255, y #keras.utils.to_categorical(y, num_classes=self.n_classes)

    def __len__(self):
        'Denotes the number of batches per epoch'
        return int(np.floor(len(self.list_IDs) / self.batch_size))


    def __getitem__(self, index):
            'Generate one batch of data'
```

```
        # Generate indexes of the batch

        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]

        # Find list of IDs
        list_IDs_temp = [self.list_IDs[k] for k in indexes]

        # Generate data
        X, y = self.__data_generation(list_IDs_temp)

        return X, y
def create_model():
   vgg_model = keras.applications.VGG16(weights='imagenet',
                            include_top=True)

   layer_dict = dict([(layer.name, layer) for layer in vgg_model.layers])

   x = layer_dict['fc2'].output
   x = Dense(5, activation='softmax')(x)

   custom_model = Model(vgg_model.input, x)

   # Make sure that the pre-trained bottom layers are not trainable
   for layer in custom_model.layers[:-3]:
      layer.trainable = False

   custom_model.compile(optimizer=keras.optimizers.Adam(learning_rate=1e-3,
beta_1=0.9, beta_2=0.999, amsgrad=False),
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])
   return custom_model
dim = (224,224)
EPOCHS = 10
batch_size = 64
n_classes = 5
n_channels = 3
shuffle = True
EARLY_STOPPING = 3

training_generator = Data_generator( train_df.id, train_df.target,
batch_size=batch_size, dim=dim, n_channels=n_channels, n_classes=n_classes,
shuffle=shuffle)
```

```
validation_generator = Data_generator(test_df.id, test_df.target,
batch_size=batch_size, dim=dim, n_channels=n_channels, n_classes=n_classes,
shuffle=shuffle )
custom_model = create_model()

ES = keras.callbacks.EarlyStopping(monitor='accuracy',
patience=EARLY_STOPPING, verbose=False, mode='auto',
restore_best_weights=True)

Check = keras.callbacks.ModelCheckpoint(  'model.hdf5', monitor='val_loss',
verbose=0, save_best_only=True, save_weights_only=False, mode='auto',
save_freq='epoch' )
history = custom_model.fit_generator(generator=training_generator,
                    validation_data=validation_generator,
                    use_multiprocessing=True,
                   epochs=EPOCHS,
                    callbacks = [ES, Check],
                    workers=-1)
```

ДОДАТОК А ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДОПОВІДІ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Дипломна робота

на тему: «Методи машинного навчання в задачах розпізнавання і класифікацій особливостей документів»

Виконав :
Студент IV курсу, групи КА-61
Мрозек Євген Романович
Керівник:
Д.ф.м.н., доц. Ігнатенко О. П.

Київ, 2020

1

Мета роботи: навчити нейронну мережу для класифікації з прийнятною точністю при мінімальних об'ємах даних.

Актуальність теми пов'язана з тим, що кількість і типів створених документів росте з кожним днем. Тому необхідні моделі які будуть автоматично перевіряти їх правильність.

Об'єкт дослідження: скан-копії документів при укладенні договорів

Предмет дослідження: нейронна мережа для розпізнавання і класифікацій особливостей документів. В якості підходу навчання нейронної мережі використовується передавальне навчання(transfer learning).

2

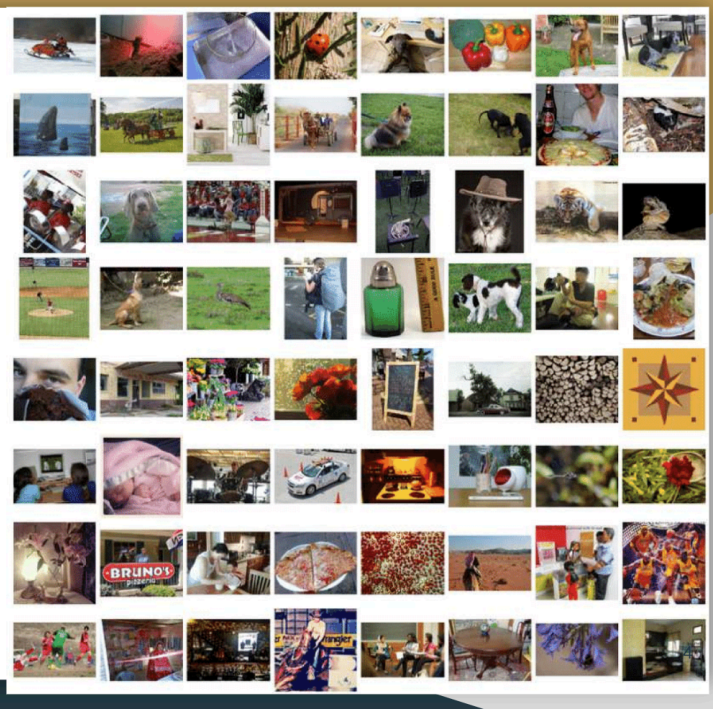
Постановка завдання

- огляд найсучасніших архітектур NN для задачі класифікації;
- огляд технології передавального навчання;
- підготовка маленького набору даних;
- порівняння ефективності звичайного підходу CV та передавального навчання для класифікації документів;

3

ImageNet

Найбільший датасет ручної розмітки в 14 мільйонів картинок, який складається з більше 20 тис класів
Приблизно 700 представників на кожен клас.
Будемо використовувати нейронні мережі навчені на цьому датасеті



Критерій порівняння

Функція втрат - Категоріальна кросс ентропія

$$CCE = -\sum_{i}^{n} y_i * \log(\tilde{y}_i)$$

y_i – справжні значення

$\tilde{y}_i$ – прогнозовані значення

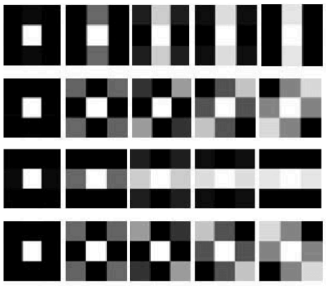
Основна метрика порівняння моделей: Точність

$$Acc = \frac{N_{correct}}{N_{All}}$$

5

Передавальне навчання

Більшість навчених NN мають одну спільну особливість: перших кілька шарів зазвичай фільтри Gabora або Blob . Це не пов'язано з завданням і використовуються на великій різноманітній кількості датасетів.
Останні шари унікальні і можуть використовуватись лише для свого конкретного завдання



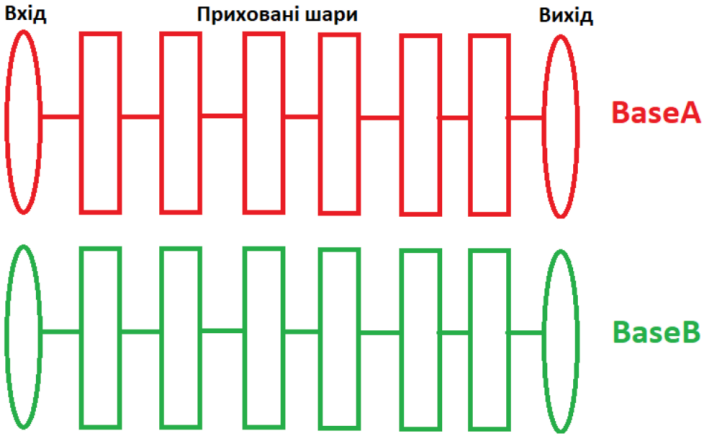
Якщо б ми знали чи перехід відбувається в межах одного шару, чи на протязі кількох - це допомогло б нам зменшити кількість необхідних даних та потужностей для навчання NN

6

Звичайний підхід

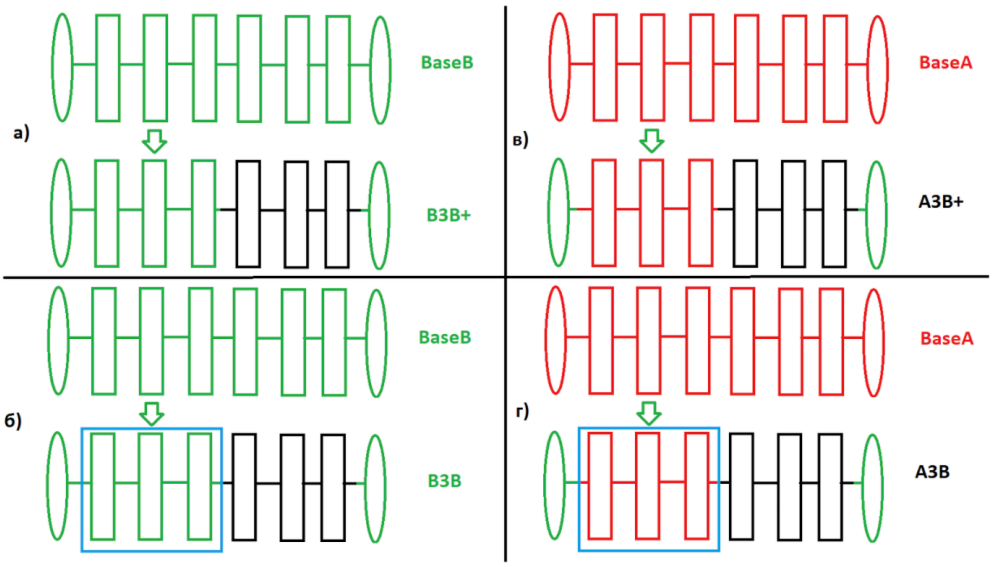
Доменна область D_a для завдання T_a

Доменна область D_b для завдання T_b



7

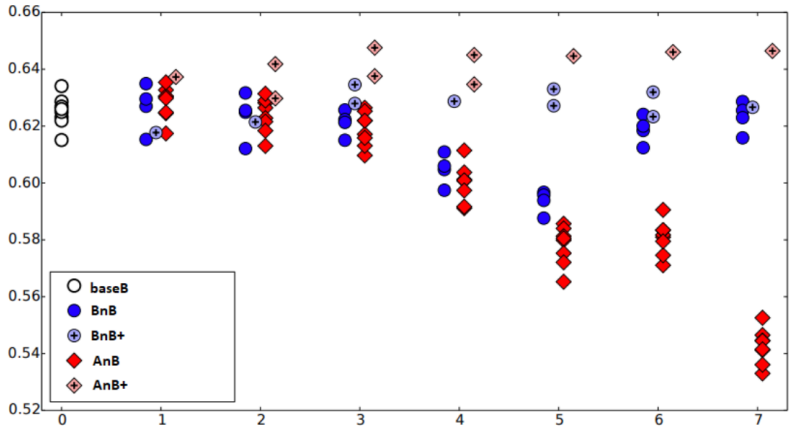
Підходи передачі коефіцієнтів



8

How transferable are features in deep neural networks?

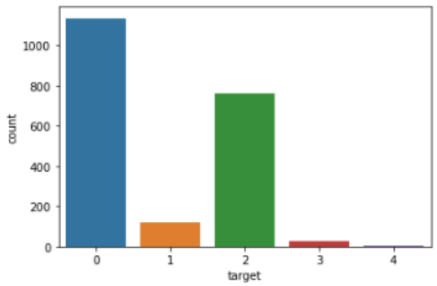
Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson

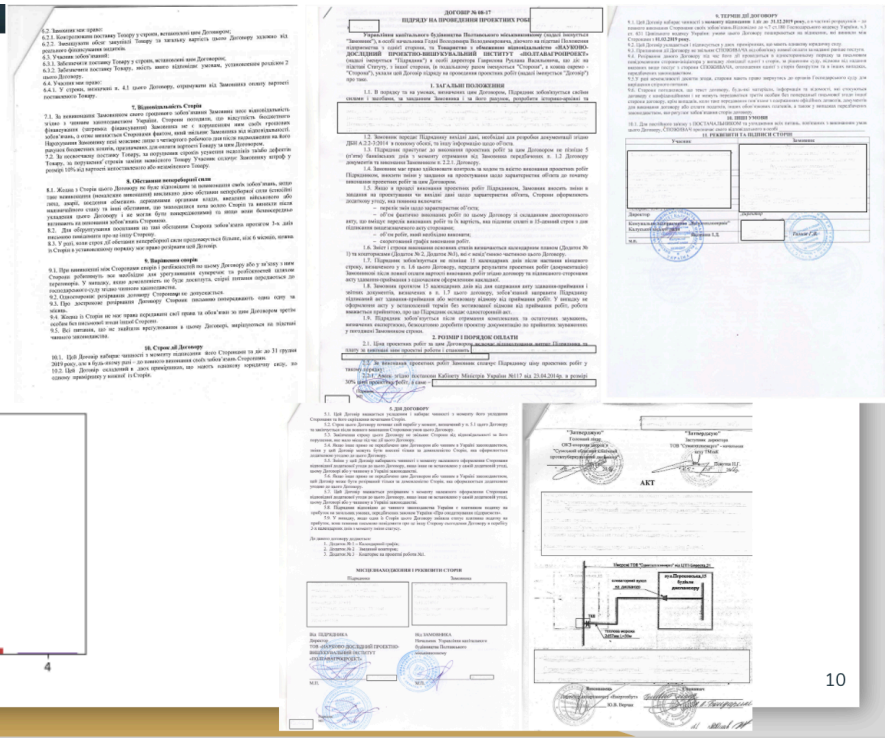


9

Набір даних

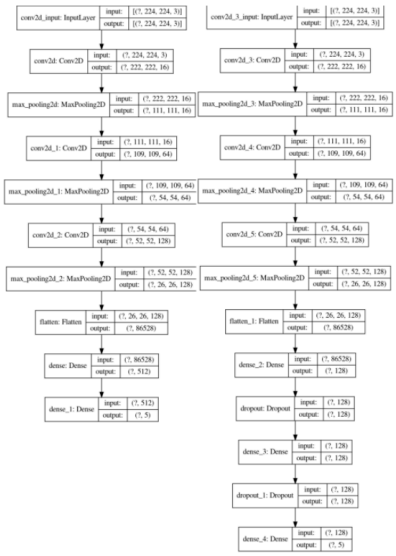
Загалом 2045 сторінок документів

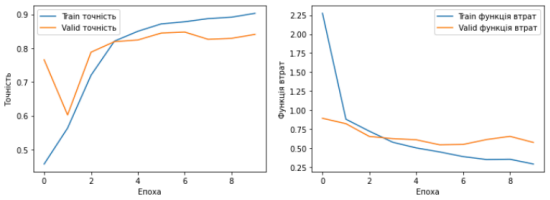




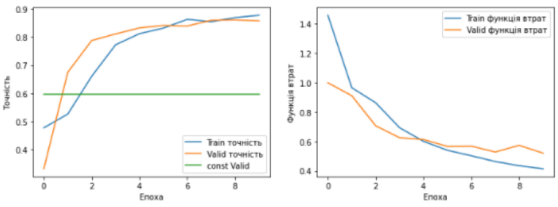
10

Звичайний підхід





Найкраща точність на валідації 0.84765625 на 7 епосі



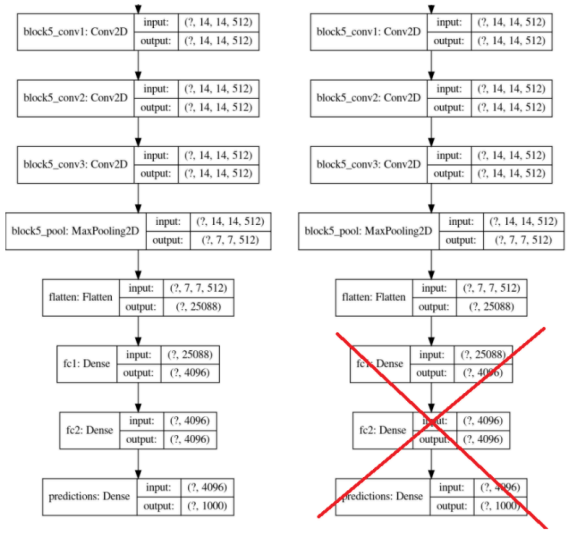
Найкраща точність на валідації 0.8613281 на 7 епосі

11

Виділення ознак

Так як ми розглядаємо вже претренировані моделі - ми можемо використати їх для виділення ознак

Далі будемо взаємодіяти не з зображенням напряму розмірності (224,224,3) а у випадку з VVG16 вектором розмірністю 25088. А на основі цього вектора робимо класифікацію класичними методами машинного навчання, як логістична регресія та градієнтний бустинг на деревах.

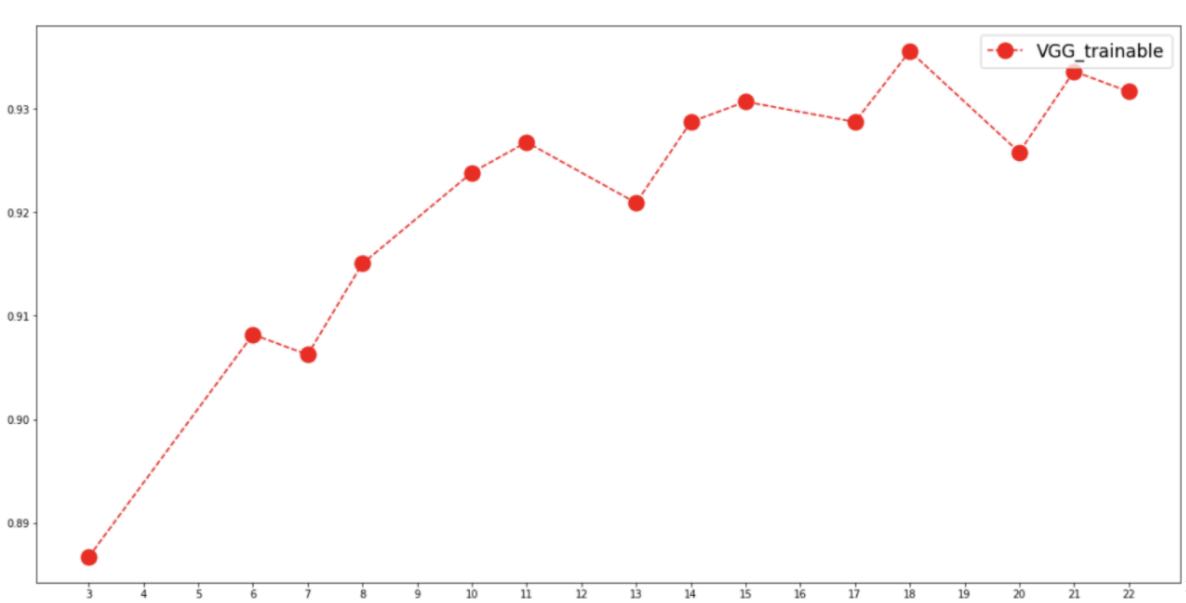


12

Арітектрура/ Модель	LGBM	LogReg	Фінальна модель
VGG16	0.8958	0.9045	0.9045
VGG19	0.8948	0.8977	0.8977
ResNet50	0.8813	0.8862	0.8862
InceptionV3	0.7984	0.7512	0.7984
DenseNet121	0.8244	0.7675	0.8244
MobileNet	0.8804	0.8929	0.8929
MobileNetV2	0.8273	0.8322	0.8322
Найкращий по методу	0.8958	0.9045	0.9045

13

Скільки шарів навчалось у VGG16

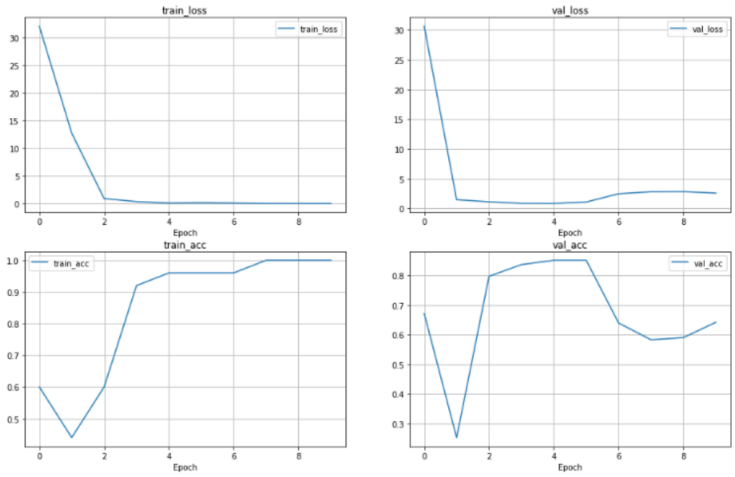


14

One/Few shot learning

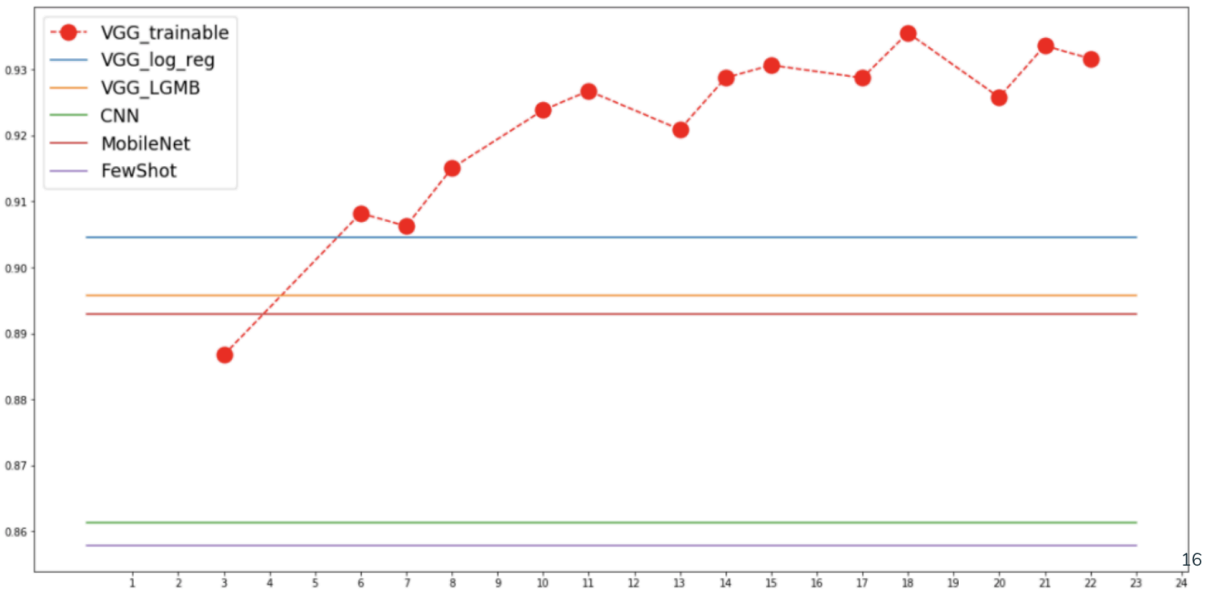
Train - 25 , Validation - 405

Найкраща точність - 0.8558519



15

Порівняння підходів звичайного підходу і з застосуванням передавального навчання



16

Висновок

- Оглянуті найсучасніші архітектури для задачі класифікації
- Досліджено технології передавального навчання
- Підготовлений маленький датасет скан копій документів
- Було порівняно звичайний підхід та з використанням передавального навчання, та як його доцільніше використовувати

17

Подальші дослідження:

- Спробувати інші сучасні архітектури, наприклад MobileNet та ResNet50, для дослідження їх шарів на загальність
- Провести аналогічний експеримент з навчальною вибіркою 20000+ елементів, щоб визначити наскільки сильно впливає відстань між датасетами
- Знайти залежність між точністю і кількістю представників в кожному класі (Few Shot learning)

18